User's Manual

Digital Gamma Finder (DGF)

PIXIE-16

Version 1.40, October 2009

XIA LLC

31057 Genstar Rd.
Hayward, CA 94544 USA

Phone: (510) 401-5760; Fax: (510) 401-5761
http://www.xia.com

Disclaimer

Information furnished by XIA is believed to be accurate and reliable. However, XIA assumes no responsibility for its use, or for any infringement of patents, or other rights of third parties, which may result from its use. No license is granted by implication or otherwise under the patent rights of XIA. XIA reserves the right to change the DGF product, its documentation, and the supporting software without prior notice.

Table of Contents

Safety

Please take a moment to review these safety precautions. They are provided both for your protection and to prevent damage to the digital gamma finder (DGF) and connected equipment. This safety information applies to all operators and service personnel.

Specific Precautions

Observe all of these precautions to ensure your personal safety and to prevent damage to either the DGF-Pixie-16 or equipment connected to it.

Do Not Hot-Swap!

To avoid personal injury, and/or damage to the DGF-Pixie-16, always turn off crate power before removing the DGF-Pixie-16 from the crate!

Servicing and Cleaning

To avoid personal injury, and/or damage to the DGF-Pixie-16, do not attempt to repair or clean the unit. The DGF hardware is warranted against all defects for 1 year. Please contact the factory or your distributor before returning items for service.

End Users Agreement

XIA LLC warrants that this product will be free from defects in materials and workmanship for a period of one (1) year from the date of shipment. If any such product proves defective during this warranty period, XIA LLC, at its option, will either repair the defective products without charge for parts and labor, or will provide a replacement in exchange for the defective product.

In order to obtain service under this warranty, Customer must notify XIA LLC of the defect before the expiration of the warranty period and make suitable arrangements for the performance of the service.

This warranty shall not apply to any defect, failure or damage caused by improper uses or inadequate care. XIA LLC shall not be obligated to furnish service under this warranty a) to repair damage resulting from attempts by personnel other than XIA LLC representatives to repair or service the product; or b) to repair damage resulting from improper use or connection to incompatible equipment.

THIS WARRANTY IS GIVEN BY XIA LLC WITH RESPECT TO THIS PRODUCT IN LIEU OF ANY OTHER WARRANTIES, EXPRESSED OR IMPLIED. XIA LLC AND ITS VENDORS DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. XIA'S RESPONSIBILITY TO REPAIR OR REPLACE DEFECTIVE PRODUCTS IS THE SOLE AND EXCLUSIVE REMEDY PROVIDED TO THE CUSTOMER FOR BREACH OF THIS WARRANTY. XIA LLC AND ITS VENDORS WILL NOT BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IRRESPECTIVE OF WHETHER XIA LLC OR THE VENDOR HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Contact Information

XIA LLC

31057 Genstar Rd.

Hayward, CA 94544 USA

Telephone: (510) 401-5760

Downloads: http://www.xia.com/DGF_Pixie-16_Download.html

Hardware Support: support@xia.com

Software Support: software_support@xia.com

1 Overview

The Digital Gamma Finder (DGF) family of digital pulse processors features unique capabilities for measuring both the amplitude and shape of pulses in nuclear spectroscopy applications. The DGF architecture was originally developed for use with arrays of multi-segmented HPGe gamma ray detectors, but has since been applied to an ever broadening range of applications.

The DGF Pixie-16 is a 16-channel all-digital waveform acquisition and spectrometer card based on the CompactPCI/PXI standard for fast data readout to the host. It combines spectroscopy with waveform digitizing and the option of on-line pulse shape analysis. The Pixie-16 accepts signals from virtually any radiation detector. Incoming signals are digitized by 12-bit 100 MSPS ADCs. Waveforms of up to 80 μs in length for each channel can be stored in a FIFO. The waveforms are available for onboard pulse shape analysis, which can be customized by adding user functions to the core processing software. Waveforms, timestamps, and the results of the pulse shape analysis can be read out by the host system for further off-line processing. Pulse heights are calculated to 16-bit precision and can be binned into spectra with up to 32K channels. The Pixie-16 supports coincidence spectroscopy and can recognize complex hit patterns.

Data readout rates through the CompactPCI/PXI backplane to the host computer can be up to 109 Mbyte/s. The standard PXI backplane, as well as additional custom backplane connections are used to distribute clocks and trigger signals between several Pixie-16 modules for group operation. A complete data acquisition and processing systems can be built by combining Pixie-16 modules with commercially available CompactPCI/PXI processor, controller or I/O modules in the same chassis.

1.1 Applications

The Pixie-16 is an instrument for waveform acquisition and MCA histogramming for arrays of gamma ray or other radiation detectors such as

- Segmented HPGe detectors.
- Scintillator/PMT combinations: NaI, CsI, BGO and many others.
- Cryogenic microcalorimeters.
- Silicon strip detectors.

1.2 Features

- Directly accepts signals from RC-type preamplifiers, photomultiplier tubes or other pulse sources.
- Two software selectable gain/attenuation settings for each analog input.
- Programmable DC-offset for each analog input.
- Digitization of 16 analog inputs in parallel at a rate of 100 MHz.
- Digital filtering with programmable filter peaking times from 0.04 to 81.28 μs (0.04 μs to 5.24 ms with alternate firmware).
- Built-in pileup inspection and pileup rejection.
- Synchronous waveform acquisition (up to 8k samples per channel) across channels and modules.

- Event processing and optional pulse shape analysis performed with 100 MHz, 32-bit floating point SHARC DSP.
- Accumulation of 16 MCA histograms (32K bins) and acquisition of event by event list mode data.
- More than 160 backplane lines for clock and trigger distribution or general purpose I/O between modules.
- Supports 32-bit, 33 MHz PCI data transfers (>100 Mbyte/second) to host computer.
- Graphical user interfaces to control and diagnose system.
- Digital oscilloscope for health-of-system analysis.
- Compact C driver libraries available for easy integration in existing user interface.

1.3 Specifications

Front Panel I/O	
Analog Signal Input	16 analog inputs. Input impedance: 1kΩ or 50 Ohm. Input amplitude ±1V pulsed, ±1.5V DC. Digitized at 100MSPS, 12-bit precision.
Digital General Purpose I/O	16 general purpose input and /or output connections: 6 inputs 3.3V LVTTL logic. 6 outputs 3.3V LVTTL logic. 4 inputs/outputs LVDS signaling. Rev-D modules have 16 LVDS pairs of Channel VETO input and 1 LVDS pair of Module VETO input
Backplane I/O	
Clock	Distributed 50 MHz clock, daisy-chained or dedicated line from slot 2.
Triggers	Two trigger busses on PXI backplane for synchronous waveform acquisition and for event triggers.
Synchronization	SYNC signal distributed through PXI backplane to synchronize timers and run start/stop to 50 ns.
Veto	VETO signal distributed through PXI backplane to suppress event triggering.
General Purpose I/O	160 bussed and neighboring lines on custom backplane to distribute hit patterns and triggers and for I/O between modules.
Host Interface	
PCI	32-bit, 33MHz Read/Write, external memory or FIFO readout rate to host over 100 MByte/s.
Digital Controls	
Gain	Two software selectable analog gains: 4 and 0.9 (by 1:0.22 attenuation). Optional +/- 10% digital gain for gain matching between channels.
Offset	DC offset adjustment from –1.5V to +1.5V, in 65536 steps.

Shaping	Digital trapezoidal filter. Rise time and flat top set independently: 0.04 to 81.28 μs (0.04 μs to 5.24 ms with alternate firmware).
Trigger	Digital trapezoidal trigger filter with adjustable threshold. Rise time and flat top set independently: 0.01 to 0.64 μs (0.01 to 40.96 μs with alternate firmware).
Data Outputs	
Spectrum	32k bins, 32 bit deep (4.2 billion counts/bin) for each channel. 256K×18bit FIFO memory for list mode data
Statistics	Real time, live time, input and output counts.
Event data	Hit pattern, pulse height (energy), timestamps, pulse shape analysis results, and waveform data.

2 Setting up

2.1 Installation

The Pixie-16 modules must be operated in a custom 6U CompactPCI/PXI chassis providing high currents at specific voltages not included in the CompactPCI/PXI standard[1]. Currently XIA provides an 8-slot and a 14-slot chassis; please inquire for further options. Put the host computer (or remote PXI controller) in the system slot (slot 1) of your chassis. Put the Pixie-16 modules into any free peripheral slot (slot 2-8 or 2-14) with the chassis still powered down. After modules are installed, power up the chassis (Pixie-16 modules are not hot swappable). If using a remote controller, be sure to boot the host computer *after* powering up the chassis.

Connect the output of your detector to the small coaxial connectors on the Pixie-16 front panels. For Rev-A modules, these connectors are of type MMCX. For Rev-B, C, and D modules, the connectors are of type SMB. The 2 mm front panel headers (and, on the Rev-B, C, and D modules, the CAT-5 connector) are for logic I/O. Preamplifier power can be connected to the DB-9 connectors on the front panel of the 6U chassis.

The Pixie-16 software includes the firmware files and DSP code files required to configure a module, Windows drivers and a Visual Basic graphical user interface. All files are included on the distribution CD-ROM and can be installed by running the installation program *Setup.exe*. Follow the instructions shown on the screen to install the software to the default folder selected by the installation program, or to a custom folder. This folder will contain 7 subfolders named *Doc*, *Drivers*, *DSP*, *Firmware*, *MCA*, *PulseShape,* and *Resources*. Make sure you keep this folder organization intact, as the interface program and future updates rely on this. Feel free, however, to add folders and subfolders for the output data at your convenience.

2.2 Getting Started

This section describes the basic steps to get initial list mode traces or MCA histograms with the Pixie-16 system. For a detailed introduction to the software interface, refer to section 3.

After installation, find the shortcut *Pixie16_VB* on your desktop and start it with a double click. You can also directly run the file *Pixie16_VB.exe* in the installation folder.

The user interface consists of a left control bar with 4 tabs: *Startup, Settings, Run,* and *Results*. The area to the right will display control panels or graphs. The top menu bar contains links to some frequently used result displays as well as some advanced parameter tables.

2.2.1 Startup

To boot the modules, select the default *Startup* tab. Enter the number of modules and specify the module's slot numbers. You can click the *Select Configuration Files* button to verify the boot files and paths are pointed to the installation folder. Then click the button *Boot Pixie-16 Modules*. You might hear several clicks from the modules as the gain relays are reset, then the

[1]Of the 5 backplane connectors available in a 6U format, the lower two are defined by the CompactPCI/PXI standard, providing basic supply voltages, PCI host I/O, and basic trigger connections. Pixie-16 modules follow this standard and are thus compatible with any CompactPCI/PXI module that uses these two connectors only. The upper three connectors are undefined in the CompactPCI/PXI standard. On Pixie-16 modules, these connectors are used for custom power supplies with high currents (1.8V, 5.5V, 3.3V) and for extended trigger distribution. Third party modules or chassis using the upper three connectors are most likely not compatible with Pixie-16 modules.

bottom status line should show a green marker indicating that the Pixie-16 modules initialized. For analysis-only operation with no modules, check the *Offline Analysis* box before booting.

2.2.2 Settings

To configure the modules for your detector, go to the *Settings* tab. Click on the *Acquire ADC Traces* button to view the input signal for either a single channel or for all 16 channels of the *Module* selected at the bottom of the panel. Click *Refresh* to acquire untriggered traces read directly from the ADC. You can adjust the sampling interval to see a longer time period. Pulses from the detector should fall in the range from 0 to 4k, with the baseline at ~400, a positive amplitude (rising edge), and no clipping at the upper limit. If the signal is not in range, click on the *Adjust Offsets* button to let the software to automatically set the DC offsets, or you could manually adjust the DC offset by clicking on the *Set DSP Parameter* button, then go to the *Analog Signal Conditioning* tab and adjust gain, offset and/or polarity.

The most critical parameter for the energy computation is the signal decay time Tau. In the *Set DAQ Parameter* panel, go to the *Decay Time* tab to set this value. You can either enter it directly for each channel, or enter an approximate value in the right control, select a channel, and click *Find it* to let the software determine the decay time automatically. Click *Accept it* to apply the found value to the channel. (If the approximate value is unchanged, the software could not find a better value.)

When the signal is in range and the decay time found, you can store the parameters on file using the *Save Parameter* button. Make sure to check the box for *DSP settings* to not only save the GUI settings (such as slot numbers), but also the DSP settings for each module (gain, offset, decay times, etc.)

2.2.3 Run

When the DSP settings have been found, at least initially, you can go to the *Run* tab to start a test data acquisition. Using the *Run Type* control, specify a list mode run to acquire waveforms and MCA histograms or an MCA mode run to only acquire histograms only. Click *Start* to begin acquisition.

2.2.4 Results

After the run, statistics, spectra or traces can be viewed by selecting the corresponding item from the *Results* tab. The data is also saved to files that can be imported in other analysis software. To obtain the best energy resolution in the spectra, it is likely necessary to refine the DSP settings as described in section 3.3.

3 Navigating the Pixie-16 User Interface

3.1 Overview

The Pixie-16 graphical user interface (Figure 3.1) provides the user a simple tool to control the Pixie-16 cards. It was written using Microsoft's Visual Basic programming language and its underlying function calls are directed to two dynamic link library (DLL) files, Pixie16AppDLL.dll and Pixie16SysDLL.dll. Those users who are interesting in learning more about these DLLs can read the Programmer's Manual.

After installing the Pixie-16 software on the user's computer, the Pixie-16 user interface can be launched by double clicking the shortcut *Pixie16_VB* on the desktop (or the executable file Pixie16_VB.exe in the installed folder). The user interface consists of a work area where DAQ graphs, tables and panels are to be shown, a control bar to the left which contains four tabs with control buttons (*Startup, Settings, Run,* and *Results*), and status indicators at the bottom. Below we describe the steps of using this interface.

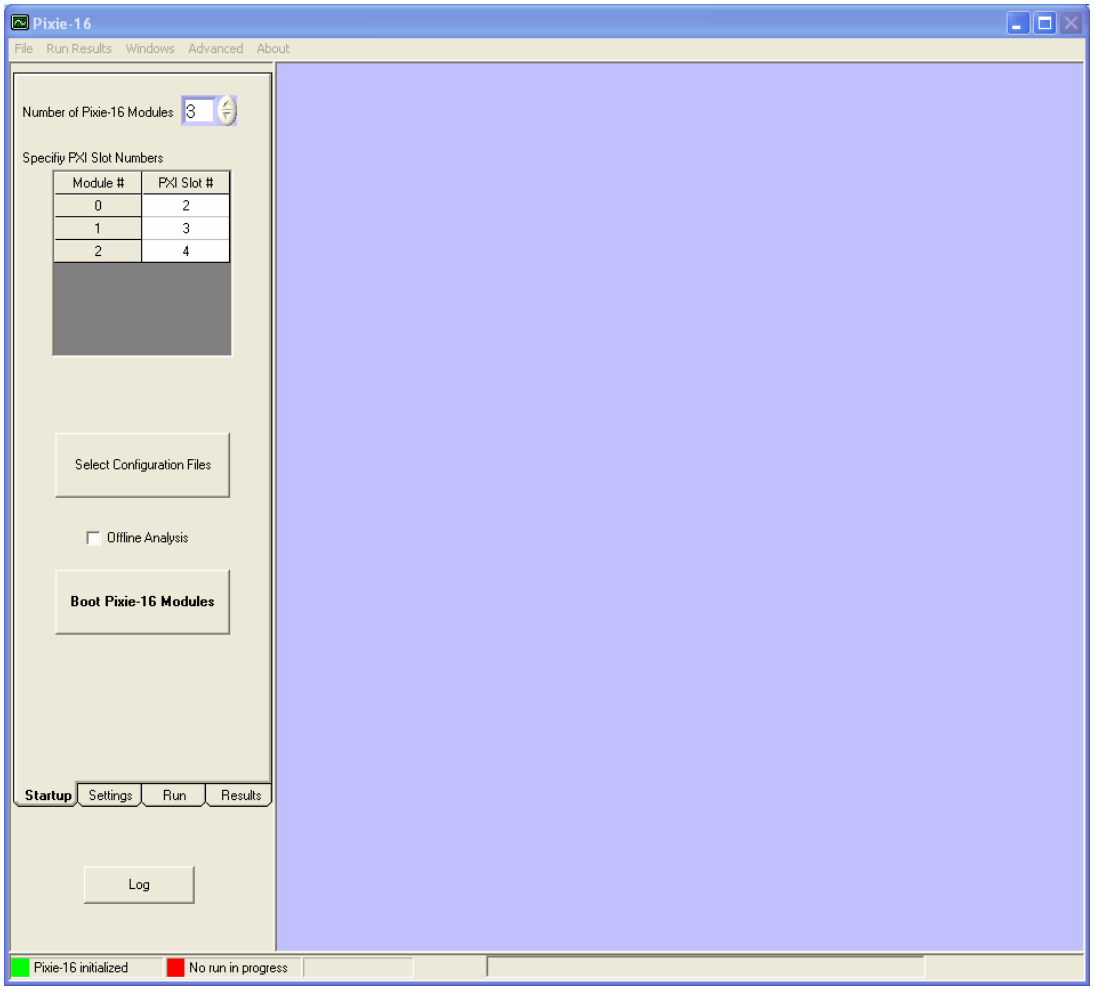

Figure 3.1: The graphical user interface for Pixie-16.

3.2 Startup

After the user interface is launched, the user will see the window as shown in Figure 3.1, with the default *Startup* tab selected in the control bar. Enter the number of modules and specify the module's slot numbers as labeled on the chassis. You can click the *Select Configuration Files* button to open the *Files and Paths* panel (Figure 3.2) and verify that the boot files and paths are pointed to the installation folder. You can directly input the file name in the boxes, or use the "file open" icon at the right end of each line to locate a specific file.

Usually, users need only change the *DSP par file* to load alternative settings (same as *Loading Parameters* in the *Setting* tab) or change the *Output Data Paths* to direct the output data into a custom location. However, if you receive code updates or custom code from XIA, you can select here which code to use.

TIP: changing the *Boot files path* will automatically change the file paths for all files.

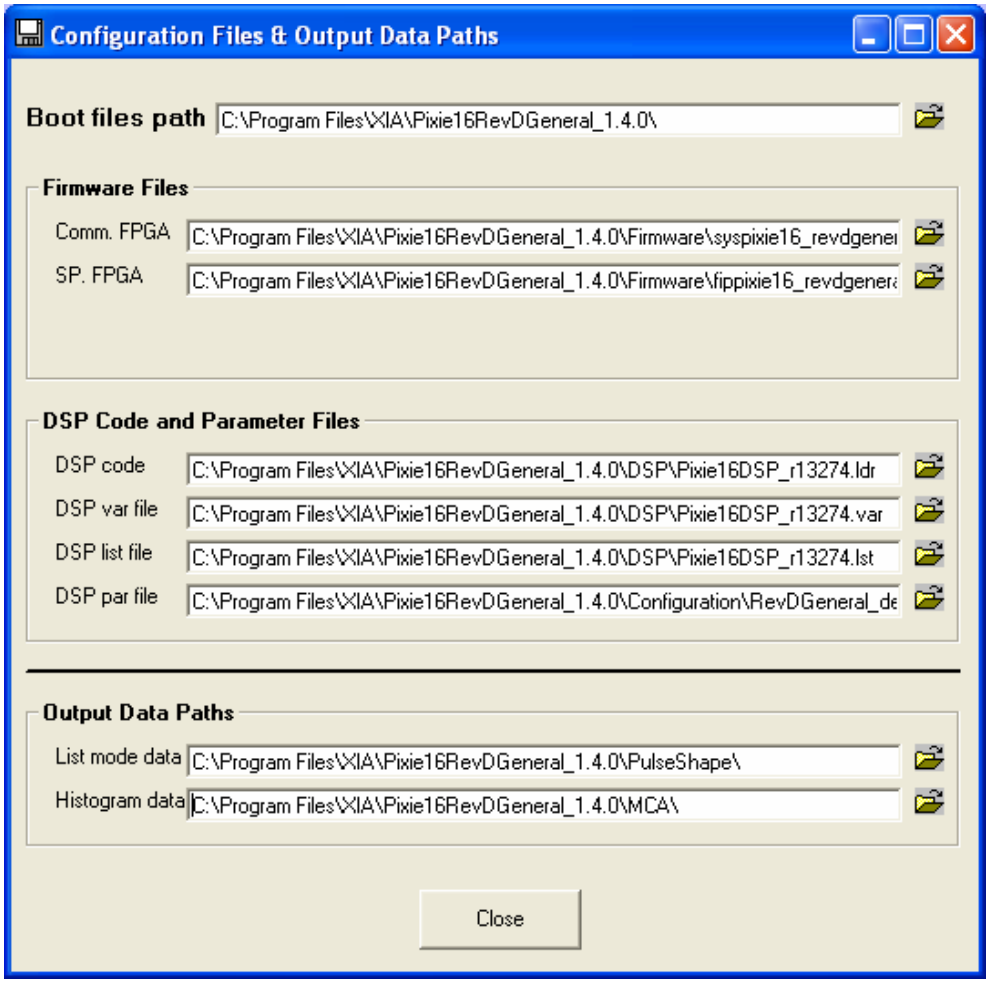

Figure 3.2: The boot files and path selection panel.

When the files and paths are set correctly, click the button *Boot Pixie-16 Modules*. You should hear several clicks from the modules as the gain relays are reset, then the bottom status line should show a green marker indicating that the Pixie-16 modules initialized. If one or more cards failed to boot, click the *Log* button to view a series of diagnostic messages. The messages are

also stored in a file called "Pixie16msg.txt" and can be sent to XIA for support. It is located in the same folder as the user interface program Pixie16_VB.exe.

For analysis-only operation with no modules, check the *Offline Analysis* box before booting. In offline mode, the user can still access every button or control of the interface. You can view results from previous acquisitions by loading the result files.

3.3 Settings

The operation of the Pixie-16's on-board DSP is controlled by a number of parameters. They can be set using the *Set DAQ Parameters* panel, opened by clicking on *Set DSP Parameters* in the *Settings* tab. The panel has 12 tabs, as shown in Figure 3.3. Using the button in the left control bar, parameters can be

Copied from one channel to some or all channels and modules in the system. When copying, first select source module and channel at the top of the copy panel, then select the items to copy on the left (corresponding to the 12 tabs of the *Set DAQ Parameters* panel), then select the destination channels and modules, and finally click on *Copy*.

Saved to disk. When saving, make sure to check the box for *DSP settings* to not only save the GUI settings (such as slot numbers), but also the DSP settings for each module (gain, offset, decay times, etc).

Loaded from disk.

3.3.1 Filter

The *Filter* tab shows the settings for the energy filter used to compute the pulse height and of the trigger filter to detect pulses. The filtering principle is described in section 6. General rules of thumb for the following important parameters are

1. The *energy filter flat top time* should be about the same as the pulse rise time.
2. The *energy filter rise time* can be varied to balance resolution and throughput. Typically, energy resolution increases with the length of the filter rise time, up to an optimum when longer filters only add more noise into the measurement. The filter dead time is about $TD = 2 \times (T_{rise} + T_{flat})$, and the maximum throughput for Poisson statistics is 1/(TD*e). For HPGe detectors, a rise time of 4-6μs is usually appropriate.
3. A longer *trigger filter rise time* averages more samples and thus allows setting lower thresholds without triggering on noise.
4. Typically the *threshold* should be set as low as possible, just above the noise level.

The remaining parameters are usually minor adjustments for fine tuning and otherwise can remain at the default values:

5. A longer *trigger filter flat top time* makes it easier to detect slow rising pulses.
6. Choose the smallest energy *filter range* that allows setting the optimum energy filter rise time. Larger filter ranges allow longer filter sums, but increase the granularity of possible values for the energy filter rise time and flat top time and increase the jitter of latching the energy filter output relative to the rising edge of the pulse. This is usually only important for very fast pulses.

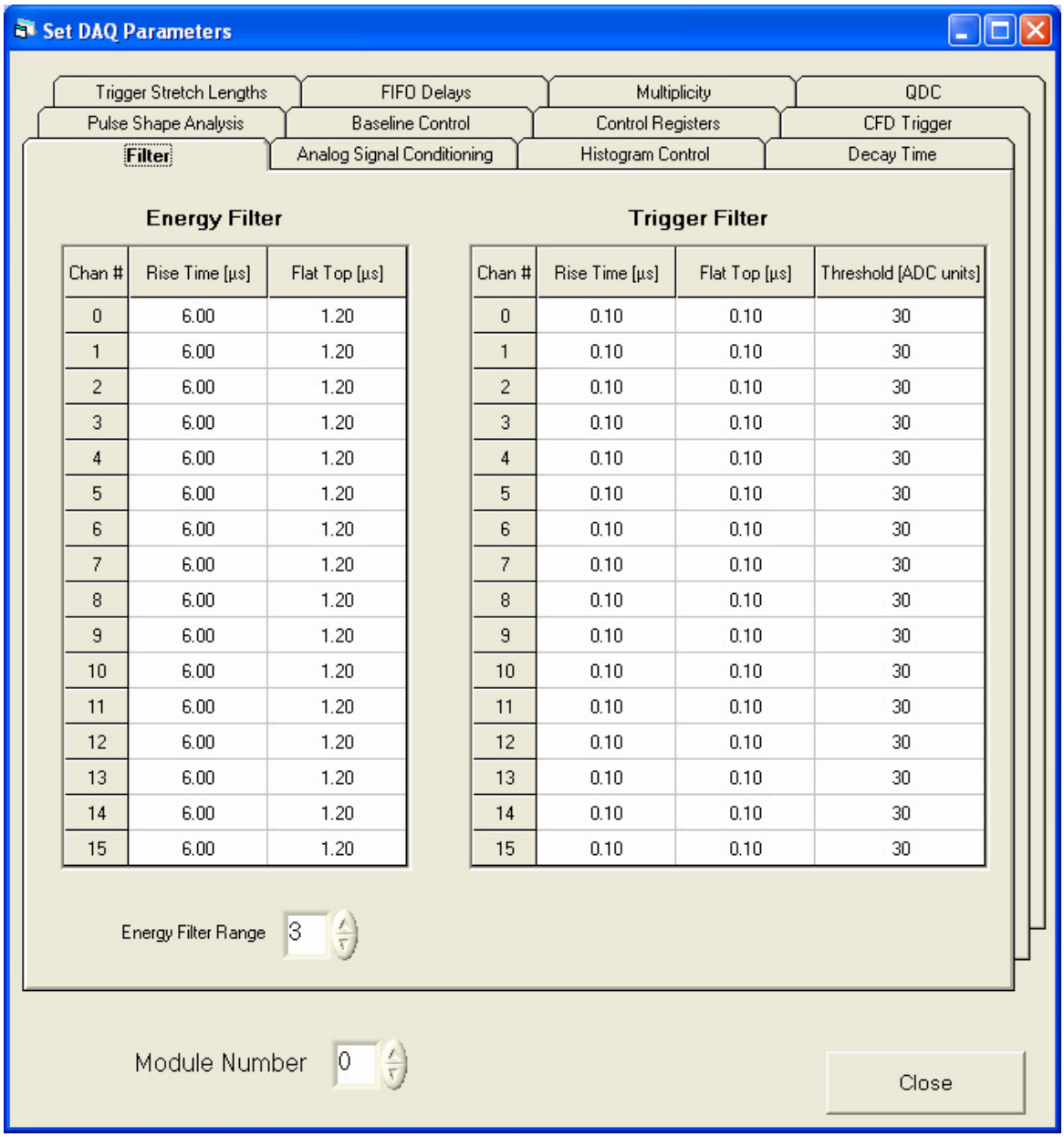

Figure 3.3: Set DAQ Parameters panel.

3.3.2 Analog Signal Conditioning & Acquire ADC Traces

The *Analog Signal Conditioning* tab controls the analog gain, offset and polarity for each channel. It is useful to click on *Acquire ADC Traces* in the left control bar to view the signal read from the ADCs while adjusting these parameters (see Figure 3.4). The display shows all 16 channels of a module in 4 graphs of 4; you can set the sampling interval for each block to capture a longer time frame. Click *refresh* to update the graph.

Pulses from the detector should fall in the range from 0 to 4095, with the baseline at ~400 to allow for drifts and/or undershoots and no clipping at the upper limit. If there is clipping, adjust the *Gain* and *Offset* or click on the *Adjust Offsets* button to let the software set the DC offsets to proper levels.

Since the trigger/filter circuits in the FPGA only act on rising pulses, negative pulses are inverted at the input of the FPGA, and the waveforms shown in the ADC trace display include this optional inversion. Thus set the channel's *Polarity* such that pulses from the detector appear with positive amplitude (rising edge).

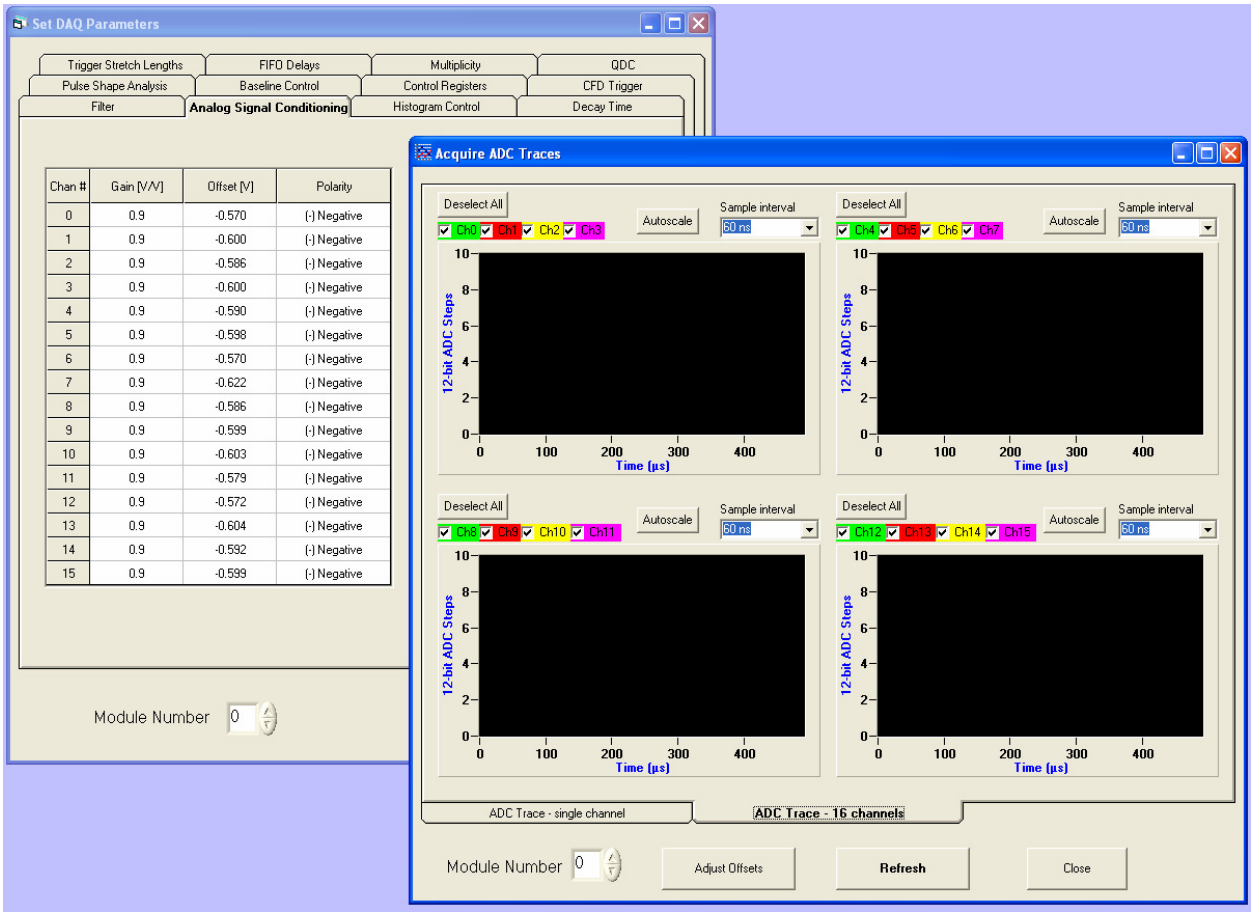

Figure 3.4: Set DAQ Parameters panel (Analog Signal Conditioning tab) and ADC trace display.

In the *single channel* tab, the ADC trace display also includes the option to view a FFT of the acquired trace. This is useful to diagnose noise contributions. Above the graph are controls for cursors and an option to change between linear and log scale. You can also save a trace to a text file using the *disk symbol* at the right.

3.3.3 Histogram Control

The *binning factor* in the *Histogram Control* tab controls the number of MCA bins in the spectrum. Energies are computed as 16 bit numbers, allowing in principle 64K MCA bins. However, spectrum memory for each channel is limited to 32K bins, so computed energy values are divided by $2^{\text{binning factor}}$ before building the histogram. *Binning Factor* is usually set to 1, but for low count rates and wide peaks, it might be useful to set it to a larger value to obtain a spectrum with fewer bins, but more counts per bin.

Emin is reserved for a future function to subtract a constant "minimum energy" from the computed energy value before binning to essentially cut off the lower end of the spectrum.

3.3.4 Decay Time

The most critical parameter for the energy computation is the signal *decay time* Tau. It is used to compensate for the falling edge of a previous pulse in the computation of the energy. You can either enter Tau directly for each channel, or enter an approximate value in the right control, select a channel, and click *Find it* to let the software determine the decay time automatically. Click *Accept it* to apply the found value to the channel. (If the approximate value is unchanged, the software could not find a better value.)

3.3.5 Pulse Shape Analysis

In the *Pulse Shape Analysis* tab, you can set the total *trace length* and the pre-trigger *trace delay* for the waveforms to be acquired in list mode runs.

3.3.6 Baseline Control & Acquire Baselines

The Pixie-16 constantly takes baseline measurements when no pulse is detected and keeps a baseline average to be subtracted from the energy filter output during pulse height reconstruction. Baseline measurements that differ from the average by more than *Baseline Cut* will be rejected as they are likely contaminated with small pulses below the trigger threshold. You can click on the *Acquire Baseline* button to view a series of baseline measurements for each channel, and in the *single channel* view you can build a histogram of baselines to verify that the *Baseline Cut* does not reject measurements falling into the main (ideally Gaussian) peak in the baseline distribution. Usually, it is sufficient to keep *Baseline Cut* at its default value.

Note: Since the baseline computation takes into account the exponential decay, no pulses should be noticeable in the baseline display if a) the decay time is set correctly and b) the detector pulses are truly exponential.

Baseline Percent is a parameter used for automatic offset adjustment; by clicking on the *Adjust Offsets* button, offsets will be set such that the baseline seen in the ADC trace display falls at the *Baseline Percent* fraction of the full ADC range (e.g. for *Baseline Percent* = 10% the baseline falls at ADC step 409 out of 4096 total).

3.3.7 Control Registers

The *Control Registers* tab sets a number of options affecting the module as a whole (Module Control Register B) or affecting each channel individually (Channel Control Register A):

1. Module Control Register B
    a) *Enable pullups for backplane bus lines*. This should be enabled for only one module in the crate.
    b) *Connect trigger signals to backplane*. You can set the module to *share triggers* over the backplane with other modules, unless you run each module (or channel) independently.
    c) *Accept external trigger and run inhibit signals*. Enable this option to let this module accept external trigger and run inhibit signals and then put the signals on the backplane so that all modules can see the same signals. This should be enabled for only one module in the crate.
    d) *Crate master module (multiple crates only).* This option is only used when multiple Pixie-16 crates communicate with each other. By enabling this option,

the mater module in each crate is responsible for sending synchronization or trigger signals to certain backplane lines. This should be enabled for only one module in the crate.

e) *Enable run inhibit signal input*. This option is only applicable to the module which has the "Accept external trigger and run inhibit signals" option enabled. This should be enabled for only one module in the crate.

f) *Multiple crates*. This option is only used when multiple Pixie-16 crates communicate with each other.

2. Channel Control Register A

a) *Good channel*. Only channels marked as good will have their events recorded. This setting has no bearing on the channel's capability to issue a trigger. There can be a triggering channel whose data are discarded. Channels not marked as good will be excluded from the automatic offset adjustment.

b) *Histogram energies*. When this box is checked, pulse height (energy) computed for each event will be incremented to an energy histogram in the MCA memory.

c) *Capture trace*. When this box is checked, trace will be captured and recorded for each event, along with other list mode information, e.g. timestamp, energy, etc.

d) *Capture QDC sums*. When this box is checked, eight QDC sums will be recorded for each event. QDC sums are consecutive sums of the list mode trace.

e) *Enable CFD trigger*. Check this box to enable this channel's CFD trigger. Otherwise, regular trapezoidal fast trigger will be used.

f) *Require global external trigger for validation*. Check this box to require global external trigger to validate events for this channel.

g) *Capture raw energy sums and baseline*. Check this box to record raw energy sums and baseline value for each event.

h) *Require channel external trigger for validation*. Check this box to require channel external trigger to validate events for this channel.

i) *Enable pileup rejection*. Check this box to enable pileup rejection for this channel. Otherwise, pulses will still be recorded even if they are piled up.

3.3.8 CFD Trigger

The following CFD algorithm is implemented in the Pixie-16s. Assume the digitized waveform stream can be represented by data series Trace[i], i=0,1,2,… First the fast filter response of the digitized waveform is computed as follows:

$$FF[i] = \sum_{j=i-(FL-1)}^{i} Trace[j] - \sum_{j=i-(2*FL+FG-1)}^{i-(FL+FG)} Trace[j] \quad (1)$$

Where FL is called the fast length and FG is called the fast gap of the digital trapezoidal filter. Then the CFD is computed as follows:

$$CFD[i+D] = FF[i+D] - FF[i] / 2^{(W+1)} \quad (2)$$

Where D is called the CFD delay length and W is called the CFD scaling factor.

The CFD zero crossing point (ZCP) is then determined when CFD[i] > 0 and CFD[i+1] < 0. The timestamp is latched at Trace point i, and the fraction time f is given by the ratio of the two CFD response amplitudes right before and after the ZCP.

$$f = \frac{CFDout1}{(CFDout1 + CFDout2)} \times 10ns \quad (3)$$

Where CFDout1 is the CFD response amplitude right before the ZCP and CFDout2 is the CFD response amplitude (absolute value since CFDout2 is negative) right after the ZCP. The Pixie-16 DSP computes the CFD final value as shown below and stored it in the output data stream for online or offline analysis.

$$CFD = \frac{CFDout1}{(CFDout1 + CFDout2)} \times 65536 \quad (4)$$

Valid CFD Scale values and corresponding CFD scaling factors are

CFD Scale	Corresponding CFD scaling factor
0	0.25
1	0.125
2	0.0625
3	0.03125

3.3.9 Trigger Stretch Lengths

External trigger stretch can be adjusted between 10 ns and 40.96 µs. It is used to stretch the global external trigger pulse.

Veto stretch can be adjusted between 10 ns and 40.96 µs. It is used to stretch the channel veto pulse.

Fast trigger backplane length can be adjusted between 10 ns and 40.96 µs. It is used to stretch the fast trigger pulse to be sent to the backplane for sharing with other modules.

3.3.10 FIFO Delays

External delay length can be adjusted between 0 and 2.56 µs. It is used to delay the incoming pulse in order to compensate the delayed arrival of the global external trigger pulse.

Fast trigger backplane delay can be adjusted between 10 ns and 1.28 µs. It is used to delay the fast trigger pulse before it is sent to the backplane for sharing with other modules.

3.3.11 Multiplicity

Coincidence can be checked within one Pixie-16 module and/or its immediate neighbors to decide whether to accept an event. To ensure maximal flexibility when specifying how coincidence is checked, a scheme for forming 16 different coincidence groups within one Pixie-16 module was implemented as shown in Figure 3.5. Fast triggers generated within each of the 16 channels of a Pixie-16 module can be distributed to its immediate neighbors through the PXI backplane. Thus a group of up to 48 fast trigger signals can be formed within one Pixie-16 module by combining all fast triggers from the module itself and its two immediate neighbors.

Furthermore, up to 16 such groups can be formed within one Pixie-16 module, and each group can have each of its 48 fast trigger signals enabled or disabled by using a user defined contribution mask (48-bit).

It should be pointed out that two neighboring modules share the 16 nearest neighbor lines between them, i.e., if for instance one module sends 5 channels' fast triggers to its left neighbor module, its left neighbor module can then only send 11 channels' fast triggers to its right neighbor module. Therefore, it should be very careful when arranging groups of multiplicity through transferring fast triggers to neighboring modules to ensure that there will be no bus contention on the backplane.

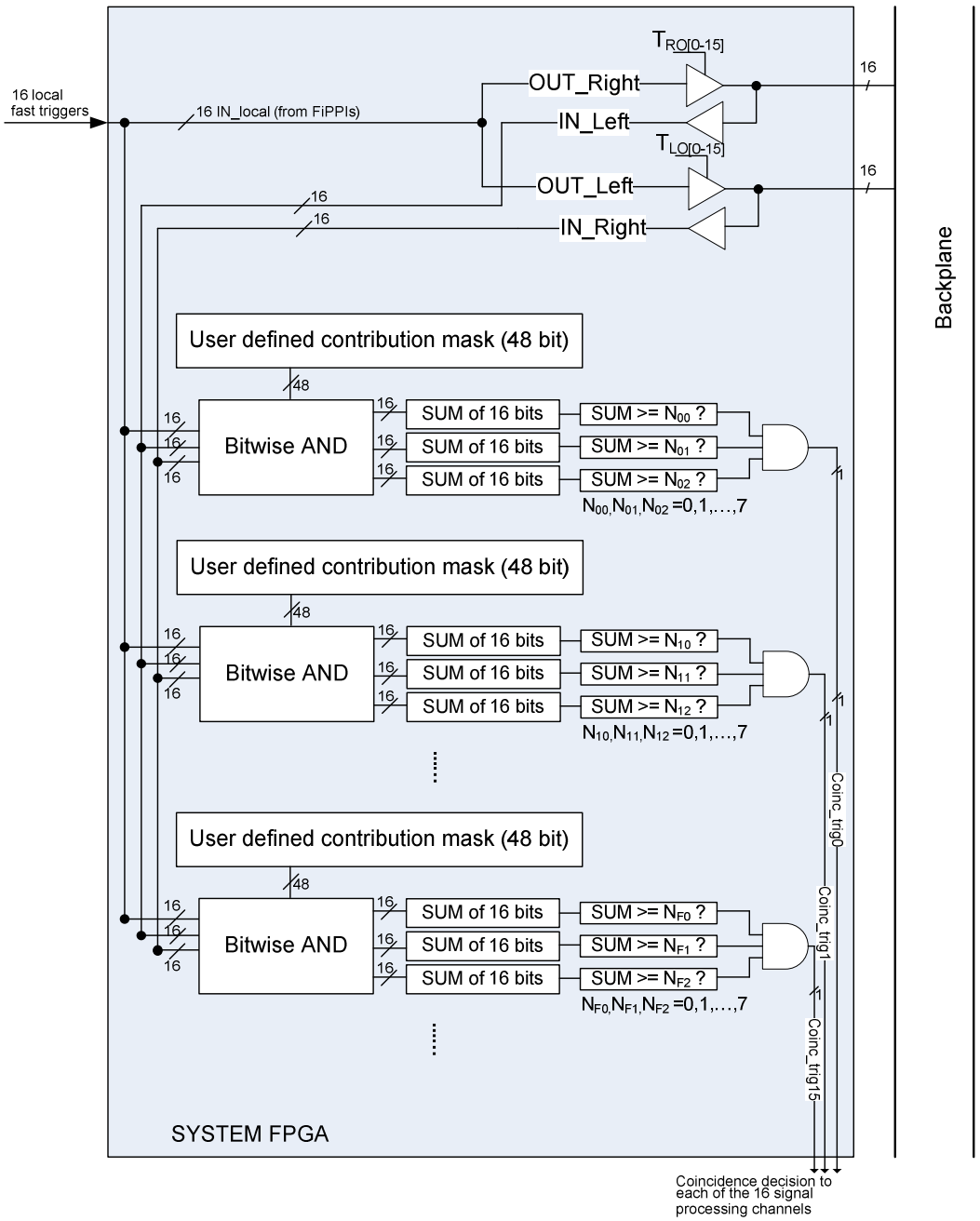

Figure 3.5: Generation of groups of coincidence triggers.

Each group of 48 fast triggers will first be logically ANDed with the 48-bit user defined contribution mask, and then each 16-bit subgroup, i.e. 16 masked fast triggers from a Pixie-16 module, is summed up to form a 3-bit multiplicity number, i.e. a multiplicity with a value of up to 7. The 3-bit multiplicity number is then compared to a user downloadable threshold value (N_{ij} <= 7). The comparison results from the three subgroups are then ANDed and that final result is sent to that given signal processing channel where the coincidence decision is desired. The following table listed the value of multiplicity output at different multiplicity threshold. SUM is the summation of the 16 individual bits of a multiplicity subgroup.

Multiplicity Threshold N_{ij} (i=0,1,2, j=0,1,…,F)	**Multiplicity Output**
0	(SUM >= 0) ? 1 : 0
1	(SUM >= 1) ? 1 : 0
2	(SUM >= 2) ? 1 : 0
3	(SUM >= 3) ? 1 : 0
4	(SUM >= 4) ? 1 : 0
5	(SUM >= 5) ? 1 : 0
6	(SUM >= 6) ? 1 : 0
7	(SUM >= 7) ? 1 : 0

The following table describes the two multiplicity parameters in the Multiplicity tab.

Multiplicity Parameters	**Description**
Multiplicity Mask Low[15:0]	User defined 48-bit contribution mask [15:0] (masking fast triggers from the module itself)
Multiplicity Mask Low[31:16]	User defined 48-bit contribution mask [31:16] (masking fast triggers from the module's right neighbor)
Multiplicity Mask High[15:0]	User defined 48-bit contribution mask [47:32] (masking fast triggers from the module's left neighbor)
Multiplicity Mask High[21:16]	Reserved
Multiplicity Mask High[24:22]	3-bit threshold for 1st 16-bit coincidence group multiplicity
Multiplicity Mask High[27:25]	3-bit threshold for 2nd 16-bit coincidence group multiplicity
Multiplicity Mask High[30:28]	3-bit threshold for 3rd 16-bit coincidence group multiplicity

3.3.12 QDC

Eight QDC sums, each of which can have different length varying from 10 ns to 327.68 μs, are computed in the signal processing FPGA of a Pixie-16 module for each channel and the sums are written to the output data stream if the user requests so. The recording of QDC sums starts at the waveform point which is Pre-trigger Trace Length earlier than the trigger point, which is either the CFD trigger or local fast trigger depending on whether or not CFD trigger mode is enabled. The eight QDC sums are computed one by one continuously, but they are not overlapping. Both the QDC sum length and the Pre-trigger trace length are set by the user. The recording of QDC sums ends when the eight intervals have passed.

3.4 Run

At the top of the Run tab is a control to specify the run type: MCA mode run (0x301) acquiring only histograms and list mode runs (0x100) acquiring histograms as well as list mode data. See section 4 for a detailed description.

For MCA runs, data acquisition continues until the specified *run time* is reached (but you can always stop it manually). In list mode runs, the on-board FIFO memory will fill up as events are acquired and therefore it has to be read out periodically. The run time here only acts as a *time out* after which the user interface aborts the run. The *polling time* is important only for list mode runs – you should poll fast enough to ensure that once the on-board FIFO memory is filled, there will be no substantial delays until the user interface notices and reads out the data. To acquire more than just a single FIFO memory full of data, you can request several *spills*. The number of spills is ignored in MCA runs.

To *synchronize* start and stop over all modules in the system, check the corresponding checkbox. All modules will start the data acquisition at the same time; and once one module stops its acquisition, all other modules will stop at the same time as well. This prevents that events are only partially acquired in different modules. To reset clock counters to zero at the beginning of the run, check the box *to synchronize clocks* also.

Data will be written to the *output file* specified, even if runs are aborted or manually stopped. The full file name will be "base name" plus a 4 digit run number. If the file already exists, the data will be appended. If the corresponding checkbox is set, the run number will be incremented automatically.

Use the *Start* button to start a run, and the *Stop* button to manually end the run before the run time or number of spills is reached.

3.5 Results

The *Results* tab shows run statistics such as live time, input and output count rate for each channel and has 2 buttons to open the *MCA spectrum* display and the *list mode* display.

In the *MCA spectrum* display (Figure 3.6), the spectrum for all 16 channels is displayed after the run. The display can also be manually updated during the run by clicking on *Refresh (read from module)*. Previous data can be viewed by clicking on *Read from file*. In the *single channel* view, one channel is displayed in detail. Clicking on the *ROI* button lets the user define a region of interest for which peak area, position and FWHM is computed (though these values should be verified in full featured analysis software). Controls to modify the behavior of the cursors and to change scaling from linear to logarithmic are located above the display.

In the *list mode* display (Figure 3.7), the captured waveforms for each channel are displayed together with the computed energy and the timestamps. You can also specify a previous file using the open file dialog. Only those channels that were set to "Good channel" will have valid energy and traces if "Capture trace" is enabled. You can scroll through the events using the *Select event number* control.

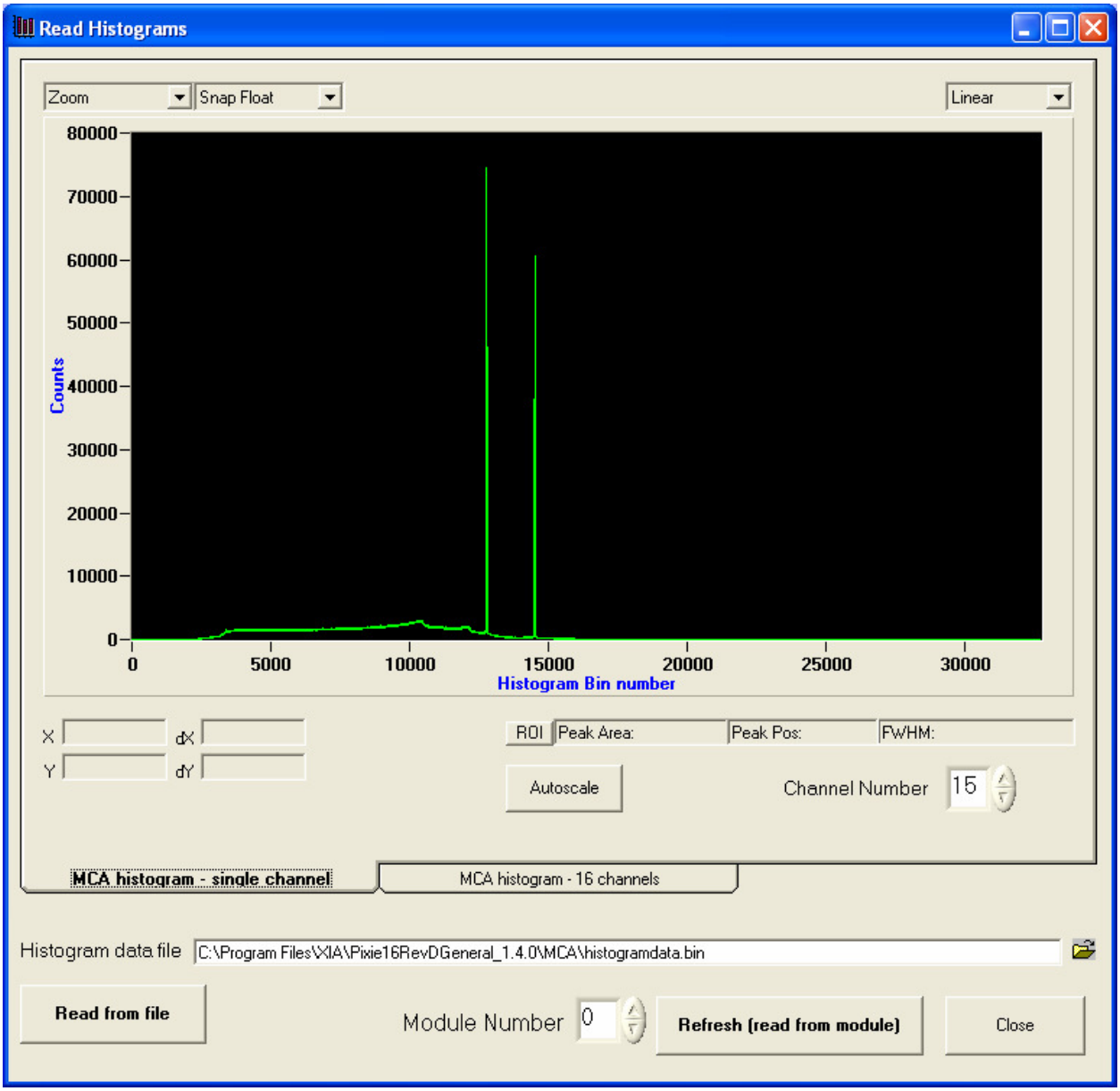

Figure 3.6: The MCA spectrum display panel.

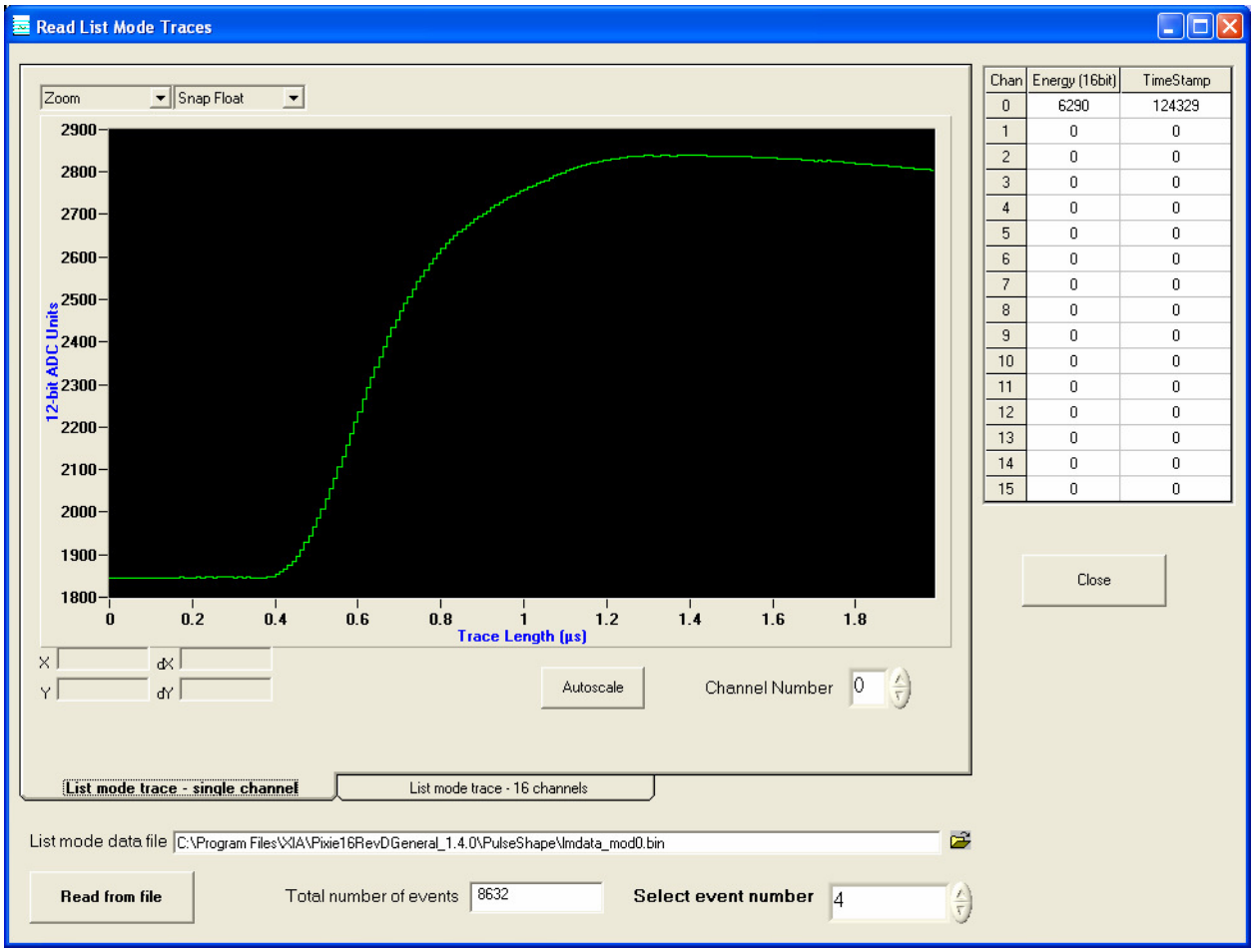

Figure 3.7: The list mode data display panel.

4 Data Runs and Data Structures

4.1 Run types

There are two run types: List mode runs and MCA histogram runs. Histogram runs only collect spectra; list mode runs acquire data on an event-by event basis.

4.1.1 Histogram Runs

If all you want to do is to collect spectra, runs should be started in histogram (MCA) mode. For each event, pulse heights are calculated and used to increment the spectrum for the corresponding channel, then discarded. Runs can continue indefinitely, or until a preset time is reached. Besides the spectrum, run statistics are available at the end of the run.

4.1.2 List Mode Runs

If, on the other hand, you want to collect data on an event-by-event basis, gathering energies, time stamps, pulse shape analysis values, and waveforms, for each event, you should operate the Pixie-16 in multi-parametric or list mode. In list mode, you still obtain histogramming of energies, e.g. for monitoring purposes. Runs will continue until a user manually stops the run. The control software also includes a timeout feature, causing the host to end runs.

4.2 Output data structures

Output data are available in two different memory blocks. The multichannel analyzer (MCA) block resides in memory external to the DSP. The list mode data is located in the external FIFO memory that can hold 256K×18-bit words. The external FIFO allows continuous readout of list mode data while a list mode run is in progress.

4.2.1 MCA histogram data

The MCA block is fixed to 32K words (32-bit) per channel, residing in the external memory. The memory can be read out via the PCI data bus at rates over 100Mbyte/s and stored to file. If spectra of less than 32K length are requested, only part of the 32K will be filled with data.

4.2.2 List mode data

List mode data consists of an event header plus trace if “Capture trace” is enabled. The event header is used to identify the event, e.g., the channel which recorded the event, the crate slot in which the module that has this channel is installed, the timestamps and energy of the event, etc. A user has the flexibility to make selections among the following event header data options. All options are controlled by individual bits in the DSP parameter CHANCSRA (Channel Control Register A in the Control Registers tab).

- Capture raw energy sums and baseline
- Capture QDC sums
- Capture raw energy sums and baseline, and QDC sums

However, for all options the following 4 words will always be included in the event header as the first 4 words.

Index	Data						Description
0	[31]	[30:17]	[16:12]	[11:8]	[7:4]	[3:0]	**Bits [3:0]** – channel number; bits [7:4] – PXI slot number; **bits [11:8]** – PXI chassis number; **bits [16:12]** – header length; **bits [30:17]** – event length; **bit [31]** – event finish code (0 – good event; 1 – piled-up event)
	Finish Code	Event Length	Header Length	CrateID	SlotID	Chan#	
1	[31:0]						Event time (lower 32-bit of the 48-bit timestamp) recorded by the signal processing FPGA (latched by either the local fast trigger or CFD trigger)
	EVTTIME_LO[31:0]						
2	[31:16]			[15:0]			**Bits [15:0]** – event time (upper 16-bit of the 48-bit timestamp); **Bits [31:16]** – CFD fractional time × 65536 (0 if CFD trigger is not enabled)
	CFD Fractional Time			EVTTIME_HI[15:0]			
3	[31:16]			[15:0]			**Bits [15:0]** – event energy; **Bits [31:16]** – trace length (0 if no trace is recorded)
	Trace Length			Event Energy			

The following sections illustrated the three different event header output options.

4.2.2.1 Recording of raw energy sums and baseline ENABLED

Index	Data	Description
4	[31:0]	Trailing energy sum
	Energy sum - trailing	
5	[31:0]	Leading energy sum
	Energy sum - leading	
6	[31:0]	Gap energy sum
	Energy sum - gap	
7	[31:0]	Baseline value
	Baseline	

4.2.2.2 Recording of QDC sums ENABLED

Index	Data	Description
4	[31:0]	QDC sum #0
	QDCSum0	
5	[31:0]	QDC sum #1
	QDCSum1	

6	[31:0]	QDC sum #2
	QDCSum2	
7	[31:0]	QDC sum #3
	QDCSum3	
8	[31:0]	QDC sum #4
	QDCSum4	
9	[31:0]	QDC sum #5
	QDCSum5	
10	[31:0]	QDC sum #6
	QDCSum6	
11	[31:0]	QDC sum #7
	QDCSum7	

4.2.2.3 Recording of raw energy sums and baseline and QDC sums ENABLED

Index	Data	Description
4	[31:0]	Trailing energy sum
	Energy sum - trailing	
5	[31:0]	Leading energy sum
	Energy sum - leading	
6	[31:0]	Gap energy sum
	Energy sum - gap	
7	[31:0]	Baseline value
	Baseline	
8	[31:0]	QDC sum #0
	QDCSum0	
9	[31:0]	QDC sum #1
	QDCSum1	
10	[31:0]	QDC sum #2
	QDCSum2	
11	[31:0]	QDC sum #3
	QDCSum3	
12	[31:0]	QDC sum #4
	QDCSum4	
13	[31:0]	QDC sum #5
	QDCSum5	
14	[31:0]	QDC sum #6
	QDCSum6	

15	[31:0]	QDC sum #7
	QDCSum7	

4.2.2.4 Recording of trace ENABLED

If trace recording is enabled, trace data will immediately follow the last word of the event header. Since raw ADC data points are 12-bit numbers, two 12-bit numbers are packed into one 32-bit word, as shown below. Since the event header could have variable length (4, 8, 12 or 16 words) depending on the selection of various output data options, the header length, event length and trace length that are recorded in the first 4 words of the event header should be used to navigate through the output data stream.

Index	Data				Description
n	[31:0]				Last word of event header which could be 4, 8, 12, or 16 words long
	Last word of event header				
n+1	[31:28]	[27:16]	[15:12]	[11:0]	Packing of ADC Data #0 and #1
	Not used	ADC Data #1	Not used	ADC Data #0	
n+2	[31:28]	[27:16]	[15:12]	[11:0]	Packing of ADC Data #2 and #3
	Not used	ADC Data #3	Not used	ADC Data #2	
…	…				

5 Hardware Description

The Pixie-16 is a 16-channel unit designed for gamma-ray spectroscopy and waveform capturing. It incorporates five functional building blocks, which we describe below. This section concentrates on the functionality aspect. Technical specifications can be found in section 1.3.

5.1 Analog signal conditioning

Each analog input has its own signal conditioning unit. The task of this circuitry is to adapt the incoming signals to the ADC. The ADC is not a peak sensing ADC, but acts as a waveform digitizer. The stages in the analog circuitry are as follows:

1. **Termination and attenuation**
   Incoming signals are terminated with 50 Ohms with no attenuation or terminated with 1K Ohm and 1:0.22 attenuation. These values may vary if you specified custom termination/attenuation when ordering the modules.

2. **Offset**
   To compensate any remaining offset, a user controlled DAC adds up to +/- 1.5V DC to the incoming signal. This stage also has a gain of 2.

3. **Nyquist**
   A subsequent opamp amplifies the signal by another factor of 2 and removes the high frequency components prior to feeding the signal into the ADC. The anti-aliasing filter, an active Sallen-Key filter, cuts off sharply at the Nyquist frequency, namely half the ADC sampling frequency. This stage also has a gain of 2.

4. **ADC**
   The ADC digitizes incoming signal at 100 MSPS with a precision of 12-bit. The ADC has an input range of 2V. If the no-attenuation input option is used, the dynamic range of the ADC is thus ~500mV. For 1:0.22 attenuation, the overall gain reduces to ~0.9 and thus the dynamic range is 2.22V.

Though the Pixie-16 can work with many different signal forms, best performance is to be expected when sending the output from a charge integrating preamplifier directly to the Pixie-16 without any further shaping.

5.2 Trigger/Filter FPGAs

The data streams from the ADCs are grouped into 4 sets of four channels and fed into of a field programmable gate array (FPGA) also incorporating a FIFO memory for each channel. Using a pipelined architecture, the signals are also processed at the full ADC sampling rate, without the help of the on-board digital signal processor (DSP).

Note that the use of one FPGA for four channels allows sampling the incoming signal at four times the regular ADC clock frequency. If the channels are fed the same input signal, the Trigger/Filter FPGA can give the each ADC a clock with a 90 degree phase shift, thus sampling the signal four times in one clock cycle. Special software is required to combine the input streams into one; contact XIA for details.

The first task of the Trigger/Filter FPGA is to generate *fast triggers* on the rising edge of the pulse. To detect triggers, the output of a short trigger filter is compared to a user defined threshold. Once a trigger is generated, it can be distributed to other channels to stop waveforms synchronously and it is used internally for a number of functions such as pileup inspection.

The second task is to apply digital filtering to perform essentially the same action as a shaping amplifier. The important difference is in the type of filter used. In a digital application it is easy to implement finite impulse response filters, and we use a trapezoidal filter. The flat top will typically cover the rise time of the incoming signal and thus makes the pulse height measurement less sensitive to variations of the signal shape on the rising edge.

Third, the Trigger/Filter FPGA contains a pileup inspector. This logic ensures that if a second pulse is detected too soon after the first, so that it would corrupt the first pulse height measurement, both pulses are rejected as piled up. The pileup inspector is, however, not very effective in detecting pulse pileup on the rising edge of the first pulse, i.e., in general pulses must be separated by their rise time to be effectively recognized as different pulses. Therefore, for high count rate applications, the pulse rise times should be as short as possible, to minimize the occurrence of pileup peaks in the resulting spectra. A user can choose to reject a pulse if it is piled-up, or can disable the pileup rejection so that a piled-up pulse can still be recorded.

The forth component of the Trigger/Filter FPGA is the event data storing memory blocks. First, there is a FIFO memory block which is continuously being filled with waveform data from the ADC. Its read pointer is always positioned such that it points to the beginning of the pulse that caused the local fast trigger in the first place. The second memory block is the dual port memory (DPM) that is used to store the event header information and trace. The DSP can directly read data from these DPMs once it finds out there is data in those DPMs, and then the DSP sends the event data to the external FIFO memory.

5.3 Digital signal processor (DSP)

The DSP controls the operation of the Pixie-16, reads raw data from the Trigger/Filter FPGAs, reconstructs true pulse heights, applies time stamps, prepares data for output to the host computer, and increments spectra in the on-board memory.

The host computer communicates with the DSP, via the PCI interface, using a direct memory access (DMA) channel. The host sets variables in the DSP memory and then calls DSP functions to program the hardware. Through this mechanism all offset DACs are set and the Trigger/Filter FPGAs are programmed with their filter parameters.

The Trigger/Filter FPGAs process their data without support from the DSP, once they have been set up. The DSP polls continuously the status of the DPMs in the Trigger/Filter FPGAs. If there is new data in the DPMs, it responds with reading the required data from the Trigger/Filter FPGAs, computing the pulse height, and storing the event header and trace (if requested) in the external FIFO memory. It then goes back to the polling loop.

In this scheme, the greatest processing power is located in the Trigger/Filter FPGAs. Each of them processes the incoming waveforms from its associated four ADCs in real time and produces, for each valid event, a small set of distilled data from which pulse heights and arrival times can be reconstructed. The computational load for the DSP is much reduced, as it has to react only on an event-by-event basis and has to work with only a small set of numbers for each event.

5.4 PCI and Trigger Interface

The PCI interface through which the host communicates with the Pixie-16 is implemented in a PCI slave IC together with an FPGA. The configuration of this PCI IC is stored in a PROM, which is placed in the only DIP-8 IC-socket on the Pixie-16 board. The interface conforms to the commercial PCI standard. It moves 32-bit data words at a time.

The interface dos not issue interrupt requests to the host computer. Instead, for example to determine when data is ready for readout, the host has to poll a Control and Status Register (CSR) in the Communication FPGA.

The Communication FPGA links the PCI slave with the DSP and the on-board memory, both the MCA memory and the list mode FIFO memory. The host can read out the memory without interrupting the operation of the DSP. This allows updates of the MCA spectrum while a run is in progress and readout of list mode data from the FIFO without filling up memory.

A Communication FPGA also acts as an interface between the DSP, Signal Processing FPGAs and the backplane and it has 18 general purpose I/O connections available on the front panel. It can be configured to distribute trigger and hit pattern information from the Signal Processing FPGAs, and over the backplane to other Pixie-16 modules. In this way, coincidence or multiplicity decisions can be made to accept or not accept an event; or a cell in a 2D array can trigger its neighboring cells.

6 Theory of Operation

6.1 Digital Filters for γ-ray detectors

Energy dispersive detectors, which include such solid state detectors as Si(Li), HPGe, HgI_2, CdTe and CZT detectors, are generally operated with charge sensitive preamplifiers as shown in Figure 6.1 a). Here the detector D is biased by voltage source V and connected to the input of preamplifier A which has feedback capacitor C_f and feedback resistor R_f.

The output of the preamplifier following the absorption of an γ-ray of energy E_x in detector D is shown in Figure 6.1 b) as a step of amplitude V_x (on a longer time scale, the step will decay exponentially back to the baseline, see section 6.3). When the γ-ray is absorbed in the detector material it releases an electric charge $Q_x = E_x/\varepsilon$, where ε is a material constant. Q_x is integrated onto C_f, to produce the voltage $V_x = Q_x/C_f = E_x/(\varepsilon C_f)$. Measuring the energy E_x of the γ-ray therefore requires a measurement of the voltage step V_x in the presence of the amplifier noise σ, as indicated in Figure 6.1 b).

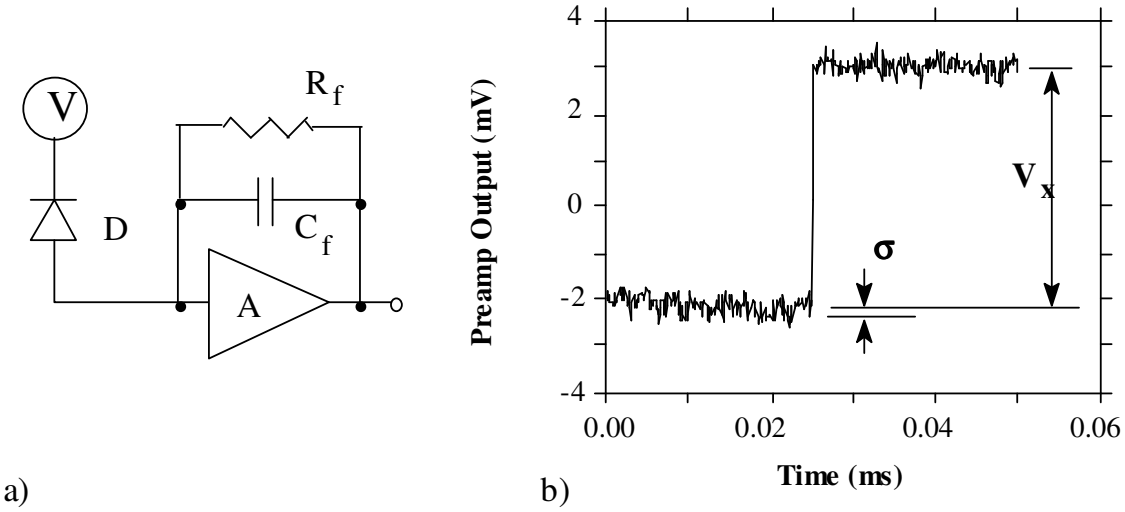

Figure 6.1: a) Charge sensitive preamplifier with RC feedback; b) Output on absorption of an γ-ray.

Reducing noise in an electrical measurement is accomplished by filtering. Traditional analog filters use combinations of a differentiation stage and multiple integration stages to convert the preamp output steps, such as shown in Figure 6.1 b), into either triangular or semi-Gaussian pulses whose amplitudes (with respect to their baselines) are then proportional to V_x and thus to the γ-ray's energy.

Digital filtering proceeds from a slightly different perspective. Here the signal has been digitized and is no longer continuous. Instead it is a string of discrete values as shown in Figure 6.2. Figure 6.2 is actually just a subset of Figure 6.1 b), in which the signal was digitized by a Tektronix 544 TDS digital oscilloscope at 10 MSA (megasamples/sec). Given this data set, and some kind of arithmetic processor, the obvious approach to determining V_x is to take some sort of average over the points before the step and subtract it from the value of the average over the points after the step. That is, as shown in Figure 6.2, averages are computed over the two regions marked "Length" (the "Gap" region is omitted because the signal is changing rapidly here), and their difference taken as a measure of V_x. Thus the value V_x may be found from the equation:

$$V_{x,k} = -\sum_{i(before)} W_i V_i + \sum_{i(after)} W_i V_i \qquad (5)$$

where the values of the weighting constants W_i determine the type of average being computed. The sums of the values of the two sets of weights must be individually normalized.

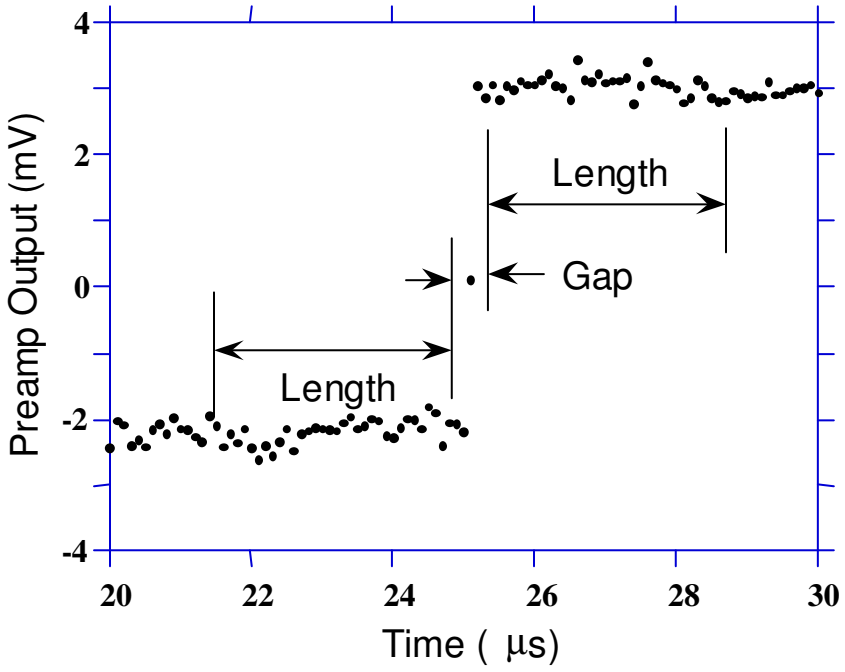

Figure 6.2: Digitized version of the data of **Figure 6.1** b) in the step region.

The primary differences between different digital signal processors lie in two areas: what set of weights $\{W_i\}$ is used and how the regions are selected for the computation of Eqn. 5. Thus, for example, when larger weighting values are used for the region close to the step while smaller values are used for the data away from the step, Eqn. 5 produces "cusp-like" filters. When the weighting values are constant, one obtains triangular (if the gap is zero) or trapezoidal filters. The concept behind cusp-like filters is that, since the points nearest the step carry the most information about its height, they should be most strongly weighted in the averaging process. How one chooses the filter lengths results in time variant (the lengths vary from pulse to pulse) or time invariant (the lengths are the same for all pulses) filters. Traditional analog filters are time invariant. The concept behind time variant filters is that, since the γ-rays arrive randomly and the lengths between them vary accordingly, one can make maximum use of the available information by setting the length to the interpulse spacing.

In principal, the very best filtering is accomplished by using cusp-like weights and time variant filter length selection. There are serious costs associated with this approach however, both in terms of computational power required to evaluate the sums in real time and in the complexity of the electronics required to generate (usually from stored coefficients) normalized $\{W_i\}$ sets on a pulse by pulse basis.

The Pixie-16 takes a different approach because it was optimized for very high speed operation. It implements a fixed length filter with all W_i values equal to unity and in fact computes this sum afresh for each new signal value k. Thus the equation implemented is:

$$LV_{x,k} = -\sum_{i=k-2L-G+1}^{k-L-G} V_i + \sum_{i=k-L+1}^{k} V_i \qquad (6)$$

where the filter length is L and the gap is G. The factor L multiplying $V_{x,k}$ arises because the sum of the weights here is not normalized. Accommodating this factor is trivial.

While this relationship is very simple, it is still very effective. In the first place, this is the digital equivalent of triangular (or trapezoidal if $G \neq 0$) filtering which is the analog industry's standard for high rate processing. In the second place, one can show theoretically that if the noise in the signal is white (i.e. Gaussian distributed) above and below the step, which is typically the case for the short shaping times used for high signal rate processing, then the average in Eqn. 6 actually gives the best estimate of V_x in the least squares sense. This, of course, is why triangular filtering has been preferred at high rates. Triangular filtering with time variant filter lengths can, in principle, achieve both somewhat superior resolution and higher throughputs but comes at the cost of a significantly more complex circuit and a rate dependent resolution, which is unacceptable for many types of precise analysis. In practice, XIA's design has been found to duplicate the energy resolution of the best analog shapers while approximately doubling their throughput, providing experimental confirmation of the validity of the approach.

6.2 Trapezoidal Filtering in the Pixie-16

From this point onward, we will only consider trapezoidal filtering as it is implemented in the Pixie-16 according to Eqn. 6. The result of applying such a filter with Length L=1μs and Gap G=0.4μs to a γ-ray event is shown in Figure 6.3. The filter output is clearly trapezoidal in shape and has a rise time equal to L, a flattop equal to G, and a symmetrical fall time equal to L. The basewidth, which is a first-order measure of the filter's noise reduction properties, is thus 2L+G.

This raises several important points in comparing the noise performance of the Pixie-16 to analog filtering amplifiers. First, semi-Gaussian filters are usually specified by a *shaping time*. Their rise time is typically twice this and their pulses are not symmetric so that the basewidth is about 5.6 times the shaping time or 2.8 times their rise time. Thus a semi-Gaussian filter typically has a slightly better energy resolution than a triangular filter of the same rise time because it has a longer filtering time. This is typically accommodated in amplifiers offering both triangular and semi-Gaussian filtering by stretching the triangular rise time a bit, so that the *true* triangular rise time is typically 1.2 times the selected semi-Gaussian rise time. This also leads to an apparent advantage for the analog system when its energy resolution is compared to a digital system with the same nominal rise time.

One important characteristic of a digitally shaped trapezoidal pulse is its extremely sharp termination on completion of the basewidth 2L+G. This may be compared to analog filtered pulses whose tails may persist up to 40% of the rise time, a phenomenon due to the finite bandwidth of the analog filter. As we shall see below, this sharp termination gives the digital filter a definite rate advantage in pileup free throughput.

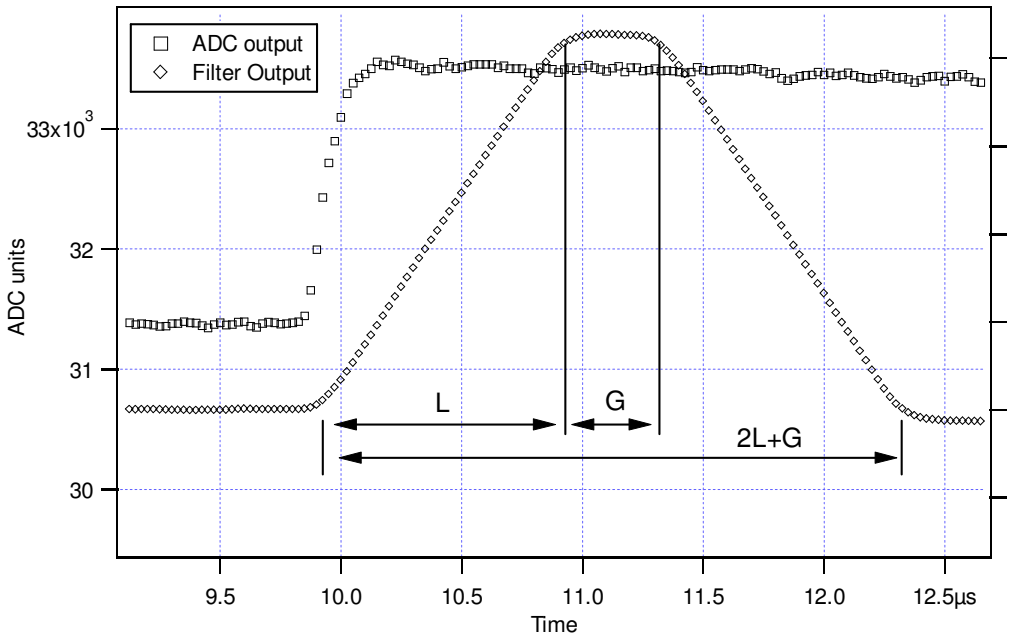

Figure 6.3: Trapezoidal filtering of a preamplifier step with L=1µs and G=0.4µs.

6.3 Baselines and preamplifier decay times

Figure 6.4 shows an event over a longer time interval and how the filter treats the preamplifier noise in regions when no γ-ray pulses are present. As may be seen the effect of the filter is both to reduce the amplitude of the fluctuations and reduce their high frequency content. This signal is termed the *baseline* because it establishes the reference level from which the γ-ray peak amplitude V_x is to be measured The fluctuations in the baseline have a standard deviation σ_e which is referred to as the *electronic noise* of the system, a number which depends on the rise time of the filter used. Riding on top of this noise, the γ-ray peaks contribute an additional noise term, the *Fano noise*, which arises from statistical fluctuations in the amount of charge Q_x produced when the γ-ray is absorbed in the detector. This Fano noise σ_f adds in quadrature with the electronic noise, so that the total noise σ_t in measuring V_x is found from

$$\sigma_t = \text{sqrt}(\sigma_f^2 + \sigma_e^2) \qquad (7)$$

The Fano noise is only a property of the detector material. The electronic noise, on the other hand, may have contributions from both the preamplifier and the amplifier. When the preamplifier and amplifier are both well designed and well matched, however, the amplifier's noise contribution should be essentially negligible. Achieving this in the mixed analog-digital environment of a digital pulse processor is a non-trivial task, however.

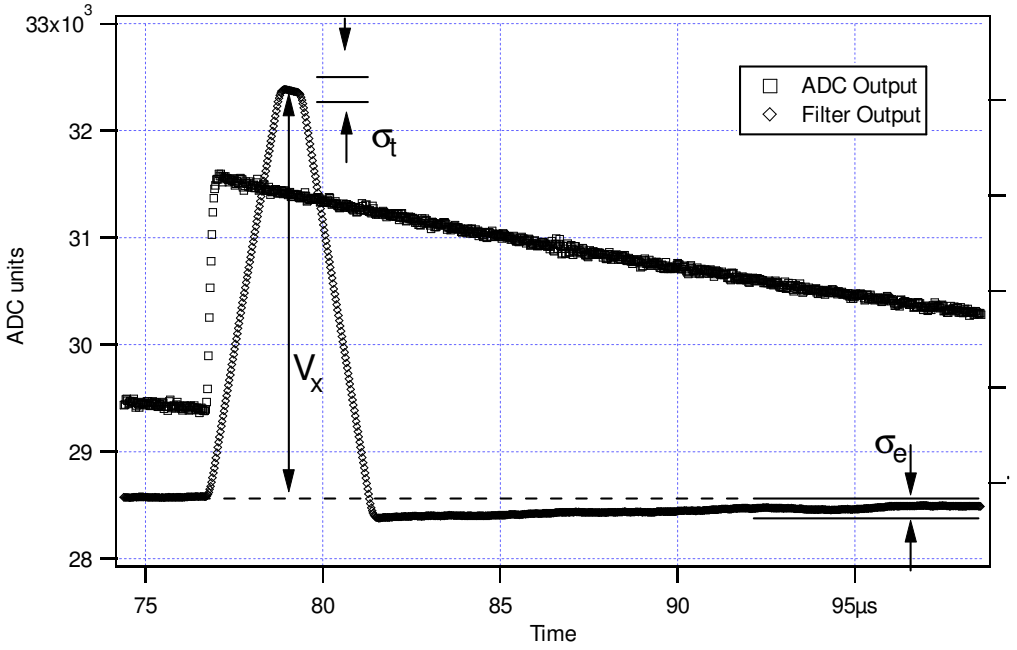

Figure 6.4: A γ-ray event displayed over a longer time period to show baseline noise and the effect of preamplifier decay time.

With a RC-type preamplifier, the slope of the preamplifier is rarely zero. Every step decays exponentially back to the DC level of the preamplifier. During such a decay, the baselines are obviously not zero. This can be seen in Figure 6.4, where the filter output during the exponential decay after the pulse is below the initial level. Note also that the flat top region is sloped downwards.

Using the decay constant τ, the baselines can be mapped back to the DC level. This allows precise determination of γ-ray energies, even if the pulse sits on the falling slope of a previous pulse. The value of τ, being a characteristic of the preamplifier, has to be determined by the user and host software and downloaded to the module.

6.4 Thresholds and Pile-up Inspection

As noted above, we wish to capture a value of V_x for each γ-ray detected and use these values to construct a spectrum. This process is also significantly different between digital and analog systems. In the analog system the peak value must be "captured" into an analog storage device, usually a capacitor, and "held" until it is digitized. Then the digital value is used to update a memory location to build the desired spectrum. During this analog to digital conversion process the system is dead to other events, which can severely reduce system throughput. Even single channel analyzer systems introduce significant deadtime at this stage since they must wait some period (typically a few microseconds) to determine whether or not the window condition is satisfied.

Digital systems are much more efficient in this regard, since the values output by the filter are already digital values. All that is required is to take the filter sums, reconstruct the energy V_x, and add it to the spectrum. In the Pixie-16, the filter sums are continuously updated by the RTPU (see section 5.2), and only have to be read out by the DSP when an event occurs. Reconstructing the energy and incrementing the spectrum is done by the DSP, so that the RTPU is ready to take new data immediately after the readout. This usually takes much less than one filter rise time, so

that no system deadtime is produced by a "capture and store" operation. This is a significant source of the enhanced throughput found in digital systems.

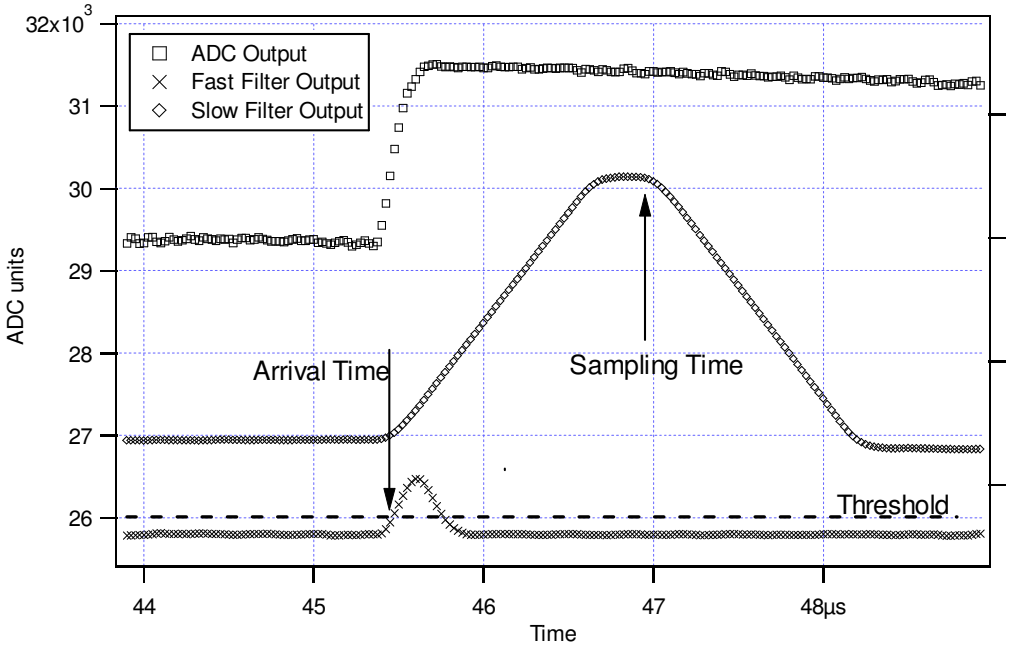

Figure 6.5: Peak detection and sampling in the Pixie-16.

The peak detection and sampling in the Pixie-16 is handled as indicated in Figure 6.5. Two trapezoidal filters are implemented, a *fast filter* and a *slow filter*. The fast filter is used to detect the arrival of γ-rays, the slow filter is used for the measurement of V_x, with reduced noise at longer filter rise times. The fast filter has a filter length L_f = 0.1µs and a gap G_f =0.1µs. The slow filter has L_s = 1.2µs and G_s = 0.35µs.

The arrival of the γ-ray step (in the preamplifier output) is detected by digitally comparing the fast filter output to THRESHOLD, a digital constant set by the user. Crossing the threshold starts a counter to count PEAKSAMP clock cycles to arrive at the appropriate time to sample the value of the slow filter. Because the digital filtering processes are deterministic, PEAKSAMP depends only on the values of the fast and slow filter constants and the rise time of the preamplifier pulses. The slow filter value captured following PEAKSAMP is then the slow digital filter's estimate of V_x.

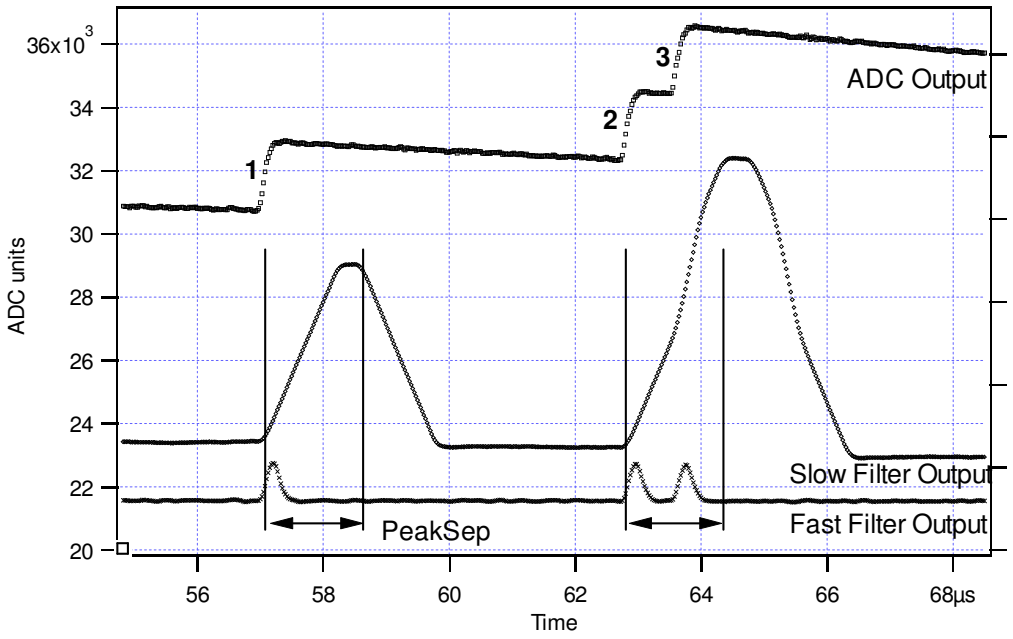

Figure 6.6: A sequence of 3 γ-ray pulses separated by various intervals to show the origin of pileup and demonstrate how it is detected by the Pixie-16.

The value V_x captured will only be a valid measure of the associated γ-ray's energy provided that the filtered pulse is sufficiently well separated in time from its preceding and succeeding neighbor pulses so that their peak amplitudes are not distorted by the action of the trapezoidal filter. That is, if the pulse is not *piled up*. The relevant issues may be understood by reference to Figure 6.6, which shows 3 γ-rays arriving separated by various intervals. The fast filter has a filter length $L_f = 0.1\mu s$ and a gap $G_f = 0.1\mu s$. The slow filter has $L_s = 1.2\mu s$ and $G_s = 0.35\mu s$.

Because the trapezoidal filter is a linear filter, its output for a series of pulses is the linear sum of its outputs for the individual members in the series. Pileup occurs when the rising edge of one pulse lies under the peak (specifically the sampling point) of its neighbor. Thus, in Figure 6.6, peaks 1 and 2 are sufficiently well separated so that the leading edge of peak 2 falls after the peak of pulse 1. Because the trapezoidal filter function is symmetrical, this also means that pulse 1's trailing edge also does not fall under the peak of pulse 2. For this to be true, the two pulses must be separated by at least an interval of L + G. Peaks 2 and 3, which are separated by less than 1.0 µs, are thus seen to pileup in the present example with a 1.2 µs rise time.

This leads to an important point: whether pulses suffer slow pileup depends critically on the rise time of the filter being used. The amount of pileup which occurs at a given average signal rate will increase with longer rise times.

Because the fast filter rise time is only 0.1 µs, these γ-ray pulses do not pileup in the fast filter channel. The Pixie-16 can therefore test for slow channel pileup by measuring the fast filter for the interval PEAKSEP after a pulse arrival time. If no second pulse occurs in this interval, then there is no trailing edge pileup. PEAKSEP is usually set to a value close to L + G + 1. Pulse 1 passes this test, as shown in Figure 6.6. Pulse 2, however, fails the PEAKSEP test because pulse 3 follows less than 1.0 µs. Notice, by the symmetry of the trapezoidal filter, if pulse 2 is rejected because of pulse 3, then pulse 3 is similarly rejected because of pulse 2.

6.5 Filter range

To accommodate the wide range of filter rise times from 0.04 μs to 81 μs, the filters are implemented in the RTPUs configurations with different clock decimation (filter ranges). The ADC sampling rate is always 10ns, but in higher clock decimations, several ADC samples are averaged before entering the filtering logic. In decimation 1, 2^1 samples are averaged, 2^2 samples in decimation 2, and so on. Since the sum of rise time and flat top is limited to 127 decimated clock cycles, filter time granularity and filter times are limited to the values are listed in Table 6.1.

Filter range	Filter granularity	max. $T_{rise}+T_{flat}$	min. T_{rise}	min. T_{flat}
1	0.02μs	2.54μs	0.04μs	0.06μs
2	0.04μs	5.08μs	0.08μs	0.12μs
3	0.08μs	10.16μs	0.16μs	0.24μs
4	0.16μs	20.32μs	0.32μs	0.48μs
5	0.32μs	40.64μs	0.64μs	0.96μs
6	0.64μs	81.28μs	1.28μs	1.92μs

Table 6.1: RTPU clock decimations and filter time granularity.

All filter ranges are implemented in the same FPGA configuration. Only the "Filter Range" parameter of the DSP has to be set to select a particular range.

For detectors with even slower time constants, the Pixie-16 can be loaded with different firmware files that perform even higher averaging. This allows pulses with decay times in the millisecond range to be processed. Contact XIA for details of this firmware.

7 Operating multiple Pixie-16 modules synchronously

When many Pixie-16 modules are operating as a system, it may be required to synchronize clocks and timers between them and to distribute triggers across modules. It will also be necessary to ensure that runs are started and stopped synchronously in all modules. All these signals are distributed through the PXI backplane.

7.1 Clock distribution

In a multi-module system there will be one clock master and a number of clock slaves or repeaters. The clock function of a module can be selected by setting shunts on Jumper JP101 at the rear of the board.

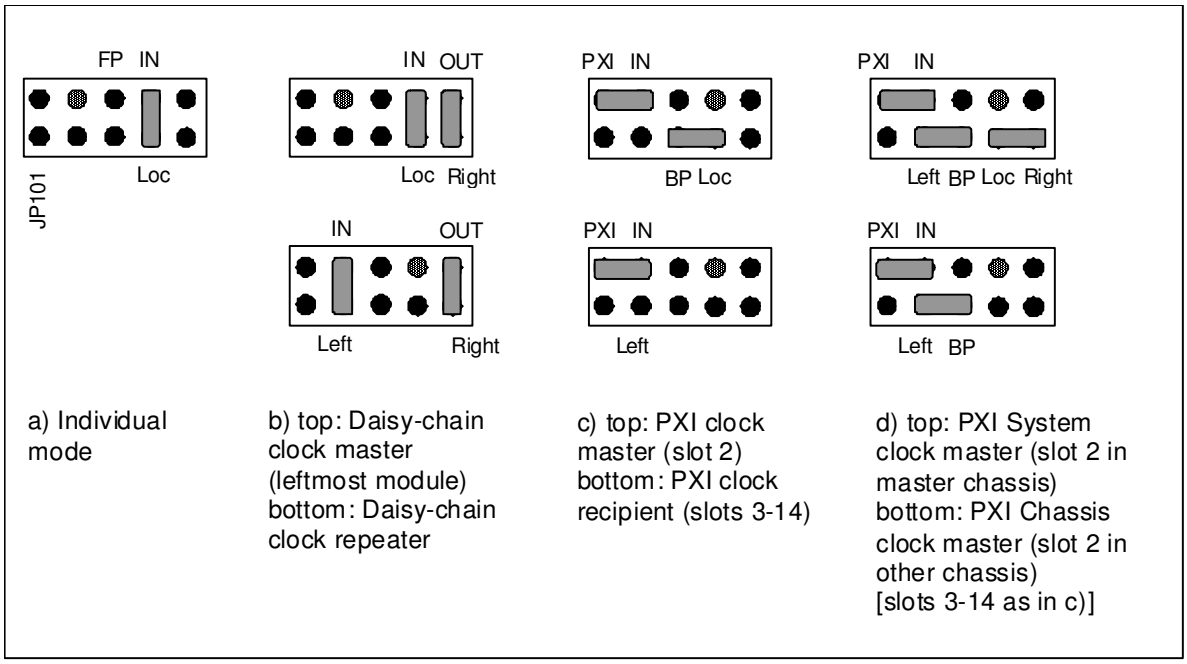

Figure 7.1: Jumper Settings for different clock distribution modes. An individual module uses its own local clock (a). In a group of modules, there will be one daisy-chained clock master in the leftmost position and several repeaters (b). Alternatively, the PXI clock distribution path can be used, with the module in slot 2 the PXI clock master and the other modules as PXI clock recipients (c). In multi-chassis systems, the module in slot 2 in the clock master chassis should be configured shown in the top picture of (d), the modules in slot 2 in all other chassis as shown in the bottom picture of (d), and modules in any other slot as in the bottom picture of (c). The local clock can be substituted by an LVDS clock input on the front panel by using the "Loc" pin instead of the "FP" pin and setting jumper JP5 to "Clk".

7.1.1 Individual Clock mode

If only one Pixie-16 module is used in the system, or if clocks between modules do not have to be synchronized, the module should be set into individual clock mode, as shown in Figure 7.1a. Connect pin 7 of JP101 (the clock input) with a shunt to pin 8 (LOC – IN). This will use the local clock crystal as the clock source.

7.1.2 Daisy-chained Clock Mode

The preferred way to distribute clocks between modules is to daisy-chain the clocks from module to module, where each module repeats and amplifies the signal. This requires one master module, located in the leftmost slot of the group of Pixie-16 modules. The master module has the same jumper settings shown in Figure 7.1b (top), using its local crystal as the input and sending its output to the right (LOC – IN, OUT – Right). Configure the other modules in the chassis as clock repeaters by setting the jumpers as shown in Figure 7.1b (bottom), using the signal from the left neighbor as the input and sending its output to the right (Left – IN, OUT – Right). There must be no gaps between modules.

7.1.3 PXI Clock Mode

A further option for clock distribution is to use the PXI clock distributed on the backplane. This clock is by default generated on the backplane (10MHz), repeated by a fan out buffer and connected to each slot by a dedicated line with minimum skew. Though the 10MHz is too slow to be a useful clock for the Pixie-16, it can be overridden by a signal from a module in slot 2.

The Pixie-16 can be configured to be the PXI clock master in slot 2, by connecting pins 6 and 8 (Loc – BP). All modules, including the clock master, should be set to receive the PXI clock by connecting pin 1 and 3 on JP101 (PXI – IN), see Figure 7.1c.

7.1.4 Multi-Chassis Clock Mode

In multi-chassis systems (Figure 7.1d), the “left” and “right” connections are taken over by a P16Trigger board on the rear backplane. The module in slot 2 in the clock master chassis should be configured to send its local clock to the P16Trigger board (Loc - Right), receive it back from the P16trigger board and send it to the PXI clock input (Left - BP), and receive the PXI clock as the input for on-board distribution (PXI – IN), see Figure 7.1d (top). The modules in slot 2 in all other chassis receive the clock from the P16trigger board and send it to the PXI clock input (Left – BP), and also receive the PXI clock as the input for on-board distribution (PXI – IN), see Figure 7.1d (bottom). The modules in any other slot should be set to use the PXI clock as the input (PXI-IN) as in Figure 7.1c (bottom).

7.2 Front panel LVDS I/O port

Figure 7.2 shows the layout of the front panel LVDS I/O port (J101). This port can be configured to either input or output 4 differential pair LVDS signals.

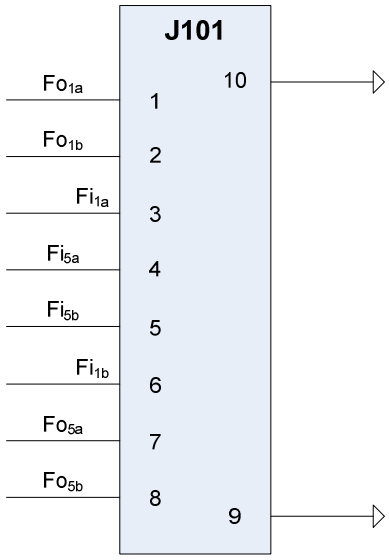

Figure 7.2: Layout of Pixie-16 front panel LVDS I/O port (J101).

7.3 Front panel digital I/O port pin

To aid system setup by a user or for debugging purpose, Pixie-16 provides up to six test pins (outputs) through the front panel connector J100. The test pins can be connected to various internal signals of the Pixie-16 to provide insight of the current status of the system. In addition, each Pixie-16 module can accept up to six external signals in TTL format. The external signals can be external triggers, external clocks, run inhibit signals, etc. Figure 7.3 shows the layout of these test pins and the following table shows the signals connected to each pin.

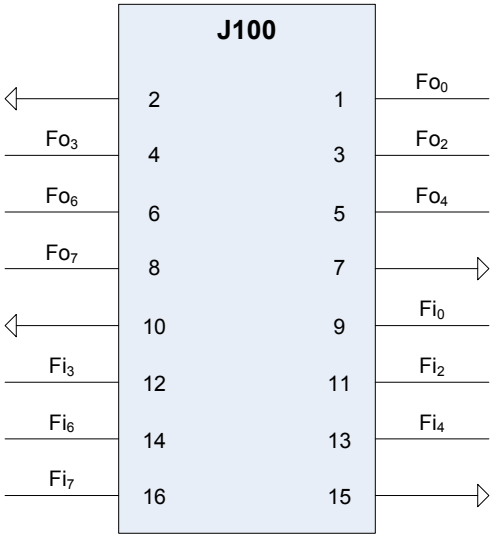

Figure 7.3: Layout of Pixie-16 front panel 3.3V I/O port (J100).

PIN #		PIN Name	Signal Name	Description
Output	1	Fo_0	FTRIG_DELAY	Delayed local fast trigger
	2	GND	GND	Ground
	3	Fo_2	FTRIG_VAL	Validated, delayed local fast trigger
	4	Fo_3	ETRIG_CE	Stretched external trigger
	5	Fo_4	VETO_CE	Stretched veto
	6	Fo_6	CHANTRIG[0]	Coincidence trigger from channel 0
	7	GND	GND	Ground
	8	Fo_7	VETO_IN[0]	Veto signal from the first channel
Input	9	Fi_0	Not used	
	10	GND	GND	Ground
	11	Fi_2	RUN_INHIBIT	Signal to inhibit run (TTL)
	12	Fi_3	Not used	
	13	Fi_4	EXT_TRIG	External trigger (TTL)
	14	Fi_6	Not used	
	15	GND	GND	Ground
	16	Fi_7	Not used	

7.4 Sample signals for front panel test pins

The timing diagrams shown below (Figure 7.4 and Figure 7.5) were acquired using a digital oscilloscope (MSO6054A) from Agilent Technologies. Shown on the top of the diagrams was the analog pulse generated by a BNC random pulser.

Figure 7.4 shows a sample timing diagram of an event validated by an external trigger. This event was validated (D1) by the existence of an external trigger (D2 and D4), but it was not vetoed since no veto pulse showed up on line D3 when delayed fast trigger (D0). On the contrary, Figure 7.5 shows a sample timing diagram of an event vetoed by the Veto signal. This event was vetoed since the stretched veto window (D3) was open when delayed fast trigger (D0) arrived. Also, even though the external trigger arrived as shown on D4, the stretched external trigger window, supposed to be shown on D2, was disabled due to the existence of the stretched veto window in D3.

Signal #	Signal Name (DIRECTOR)
1	PULSER PULSE
D0	FTRIG_DELAY
D1	FTRIG_VAL
D2	ETRIG_CE
D3	VETO_CE
D4	CHANTRIG[0]
D5	VETO_IN[0]

Figure 7.4: Sample timing diagram showing an event validated by an external trigger. This event was validated (D1) by the existence of an external trigger (D2 and D4), but it was not vetoed since no veto pulse showed up on line D3 when delayed fast trigger (D0) arrived.

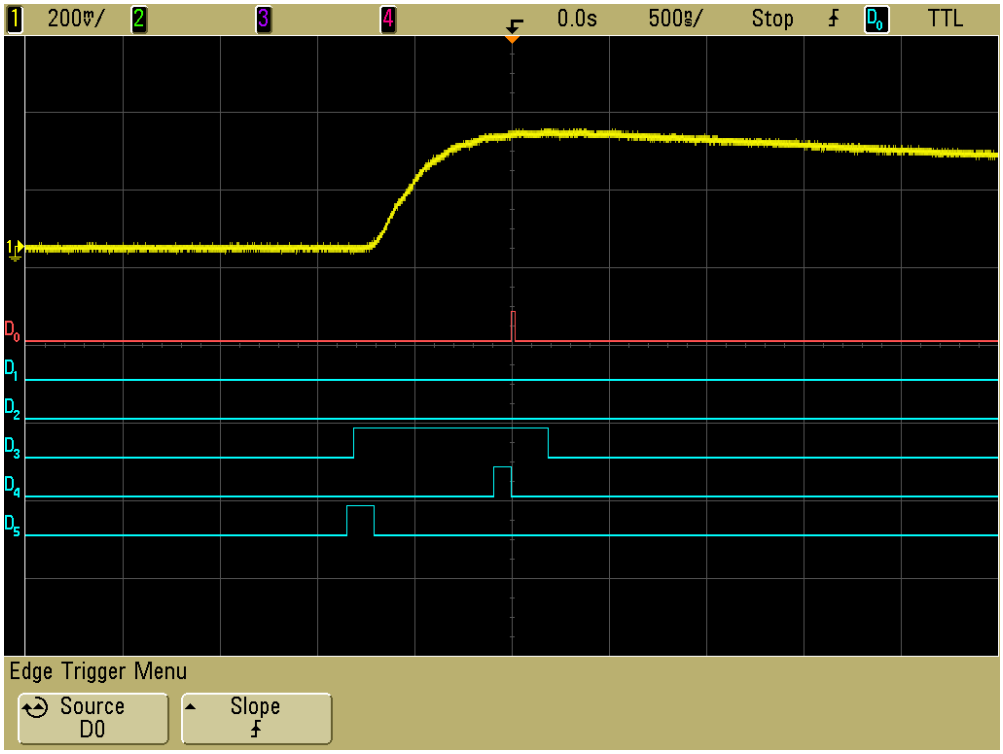

Figure 7.5: Sample timing diagram showing an event vetoed by the Veto signal. This event was vetoed since stretched veto window (D3) was open when delayed fast trigger (D0) arrived.

7.5 Trigger distribution

7.5.1 Trigger Distribution within a Module

Within a module, each channel can be enabled to issue triggers. Two kinds of triggers are distributed: First, a *Fast Trigger* indicating the trigger filter just crossed the threshold, which is used to start pileup inspection and to latch timestamps, among other things; second, a *Record Trigger* that validating the event as acceptable.

If channels are put in "group trigger" mode, each trigger enabled channel issues both kinds of triggers to the central Communication FPGA, which builds an OR of each type and sends it back to all channels. The channels then use the distributed fast triggers and record triggers instead of their own local triggers to capture data. In this way, one channel can cause data to be acquired at the same time in all other channels of the trigger group.

7.5.2 Trigger Distribution between Modules

Triggers can also be distributed over the PXI backplane. Each trigger uses a common backplane line for all modules, which is set up to work as a wired-OR. Normally pulled high, the signal is driven low by the module that issues a trigger. All other modules detect the lines being low and send the triggers to all channels (the backplane line carries a system-wide trigger that essentially acts as a 5^{th} input to the trigger OR in the Communication FPGA of each module). Alternatively, triggers can also be distributed to its nearest neighbors through the nearest neighbor lines on the backplane.

Each module can be enabled to share triggers over the backplane lines or not. In this way, a trigger group can be extended over several modules or each module can form its local sub-group.

Beside the standard trigger signals, there are more than 160 lines on the custom backplane to distribute trigger, coincidence and I/O signals. These can be configured in a number of ways. Contact XIA for customizing these lines.

7.5.3 Trigger Distribution between Chassis

Clocks and triggers can be distributed between chassis using the P16Trigger rear I/O module. See the user manual for the P16Trigger module for details.

7.6 Run Synchronization

It is possible to make all Pixie-16 modules in a system start and stop runs at the same time by using a wired-OR SYNC line on the PXI backplane. The run synchronization works as follows. When the host computer requests a run start, the DSP will first execute a run initialization sequence (clearing memory etc). At the beginning of the run initialization the DSP causes the SYNC line to be driven low. At the end of the initialization, the DSP enters a waiting loop, and allows the SYNC line to be pulled high by pullup resistors. As long as at least one of all modules is still in the initialization, the SYNC line will be low. When all modules are done with the initialization and waiting loop, the SYNC line will go high. The low->high transition will signal the DSP to break out of the loop and begin taking data. Optionally, all timers can be reset to zero when coming out of the waiting loop. From then on they will remain in synch if the system is operated from one master clock.

Whenever a module encounters an end-of-run condition and stops the run it will also drive the SYNC line low. This will be detected in all other modules, and in turn stop their data acquisition.

8 Troubleshooting

8.1 Startup Problems

The following describes solutions to common startup problems.

1. **Computer does not boot when Pixie-16 module is installed in chassis**
   This is usually caused by an incorrect clock setting on the Pixie-16 module. Each module needs to have a valid clock to respond to the computer's scanning of the PCI bus.

2. **Computer reports new hardware found, needs driver files**
   Whenever a Pixie-16 module is installed in a slot of the chassis for the first time, it is detected as new hardware, even if Pixie-16 modules have been installed in other slots previously. Point Windows to the driver files provided with the software distribution.

3. **When starting up modules in the interface program, downloads are "unsuccessful"**
   This can have a number of reasons. Verify that:
   - The files and paths point to valid locations.
   - The slot numbers entered match the location of the modules.

9 Appendix A

This section contains hardware-related information.

9.1 Jumpers

Clock Mode	JP101	PCB Reference
Single Module	Connect pins 7 - 8	LOC – IN
Daisy-Chained Clock Master	Connect pins 7 - 8, 9 - 10	LOC – IN, OUT - Right
Daisy-Chained Clock Repeater	Connect pins 3 – 4, 9 - 10	Left – IN, OUT - Right
PXI Clock Master	Connect pins 1 – 3, 6 - 8	Loc – BP, PXI – IN
PXI clock recipient	Connect pins 1 – 3	PXI – IN
Multi - Chassis System Clock Master (with P16Trigger board)	Connect pins 8 - 10, 4 - 6, 1 - 3	LOC – Right, Left – BP, PXI – IN
Multi - Chassis Chassis Clock Master (with P16Trigger board)	Connect pins 4 - 6, 1 - 3	Left – BP, PXI – IN

Table 9.1: On-board jumper settings for the clock distribution on Pixie-16 modules. Pin 1 is at the upper left. To use the front panel input instead of the local clock, connect use pin 5 instead of pin 8 and set JP 5 to "Clk".

9.2 PXI backplane pin functions

J2 pin number	PXI pin name	Connection Type	Pixie pin function
1A	LBL9	Left neighbor	Nearest Neighbor 0 (left)
16A	TRIG1	Bussed	Event Trigger
17A	TRIG2	Bussed	Veto
18A	TRIG3	Bussed	Sync
21A	LBR0	Right neighbor	Clock output
16B	TRIG0	Bussed	Fast Trigger
18B	TRIG4	Bussed	Reserved
1C	LBL10	Left neighbor	Nearest Neighbor 1 (left)
3C	LBR8	Right neighbor	Token Ring Out
18C	TRIG5	Bussed	Token Ring Return

J2 pin number	PXI pin name	Connection Type	Pixie pin function
20C	LBL0	Left neighbor	Clock input
3D	LBR9	Right neighbor	Nearest Neighbor 0 (right)
17D	STAR	Star trigger to slot 2	Reserved
2E	LBL8	Left neighbor	Token Ring In
3E	LBR10	Right neighbor	Nearest Neighbor 1 (right)
16E	TRIG7	Bussed	Bussed Clock
17E	CLK10	Clock	PXI Clock

Table 9.2: Pins of the J2 backplane connector defined in the PXI standard used by the Pixie-16. Pins not listed are not connected except for pull-ups recommended by the PXI standard.

9.3 Pinout of Digital Front Panel Connectors

Pin number	Type	Function
1	Output	
2		
3	Output	
4	Output	
5	Output	
6	Output	
7		
8	Output	
9	Input	
10		
11	Input	
12	Input	
13	Input	
14	Input	
15		
16	Input	

Table 9.3: Pinout of the 2mm front panel connector. The signal type is 3.3V LVTTL. Outputs are series terminated with 50 Ohm. Inputs have 5V tolerant Schmitt trigger buffers. Note that inputs and outputs can be daisy-chained with a split cable. Signal functions are configurable, please contact XIA for details.

Pair number	Pin	Type	Function
1	1,2	I/O or clock input	
2	3,6	I/O	
3	4,5	I/O	
4	7,8	I/O	

Table 9.4: Pinout of the CAT 5 front panel connector. The signal type is LVDS. Differential pair are terminated with 100 Ohm (between each other). Pair 1 can be sued as a clock input if JP5 is set to "Clk". Signal functions are configurable, please contact XIA for details.